Entwicklung eines AR-Systems zur Erfassung, Übertragung und Visualisierung bewegter Bilder

Masterarbeit

vorgelegt von Frederik Böhm

am 27.04.2015

Informatik Master

Fakultät Elektrotechnik & Informationstechnik

Wintersemester 2014/2015

Erster Gutachter:	Prof. Dr.-Ing. Hartwig Grabowski
Zweiter Gutachter:	Prof. Dr. sc. nat. Joachim Orb

Eidesstattliche Erklärung

Diese Master-Thesis ist urheberrechtlich geschützt, unbeschadet dessen wird folgenden Rechtsübertragungen zugestimmt:

- der Übertragung des Rechts zur Vervielfältigung der Master-Thesis für Lehrzwecke an der Hochschule Offenburg (§ 16 UrhG),
- der Übertragung des Vortrags-, Aufführungs- und Vorführungsrechts für Lehrzwecke durch Professoren der Hochschule Offenburg (§ 19 UrhG),
- der Übertragung des Rechts auf Wiedergabe durch Bild- oder Tonträger an die Hochschule Offenburg (§21 UrhG).

Hiermit versichere ich eidesstattlich, dass die vorliegende Master-Thesis von mir selbstständig und ohne unerlaubte fremde Hilfe angefertigt worden ist, insbesondere, dass ich alle Stellen, die wörtlich oder annähernd wörtlich oder dem Gedanken nach aus Veröffentlichungen, unveröffentlichten Unterlagen und Gesprächen entnommen worden sind, als solche an den entsprechenden Stellen innerhalb der Arbeit durch Zitate kenntlich gemacht habe, wobei in den Zitaten jeweils der Umfang der entnommenen Originalzitate kenntlich gemacht wurde. Ich bin mir bewusst, dass eine falsche Versicherung rechtliche Folgen haben wird.

_____

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Codeauschnittverzeichnis

Abkürzungsverzeichnis

AR	Augmented Reality
VR	Virtual Reality
KI	Künstliche Intelligenz
HMD	Head-mounted Display
App	Application (Anwendung)
MEMS	Micro Electro Mechanical System
API	Application Programming Interface
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
RTP	Real-Time Transport Protocol
JPEG	Joint Photographic Experts Group
M-JPEG	Motion-JPEG
HTTP	Hypertext Transfer Protocol
GPU	Graphics Processing Unit (Grafikprozessor)
CPU	Central Processing Unit (Prozessor)
u. a.	unter anderen

1. Einleitung

Im Rahmen dieser Arbeit wird ein Telepräsenzsystem entwickelt, bei dem ein Teleroboter mit beweglicher Kamera zum Einsatz kommt, der seine Ausrichtung über ein Head-mounted Display synchronisiert. Der Anwender des Systems sieht auf dem Display das Abbild der Umgebung des Teleroboters. Jede Kopfbewegung des Anwenders wird in Echtzeit auf die Kamerabewegung umgesetzt. Der Anwender kann sozusagen seinen Sehsinn temporär auf den Roboter auslagern, der ihn in einer entfernten Umgebung repräsentiert.

1.1 Motivation

Systeme, welche die Sinneswahrnehmungen auf einen Teleroboter auslagern, sind aus mehreren Gründen interessant. Die Künstliche Intelligenz (KI) ist noch weit davon entfernt, bei Ausführung von komplexen Aufgaben, gerade in unstrukturierten Umgebungen, einen Menschen durch einen autonomen Roboter zu ersetzen. Die kognitiven Fähigkeiten des Menschen sind in vielen Aufgabenbereichen unverzichtbar.

Die intuitive Steuerung erleichtert die Bedienung des Teleroboters und ermöglicht dem Menschen an Orten aktiv zu werden, die unerreichbar oder gesundheitsschädlich sind. Gerade in Gefahrenbereichen, wie Labore mit gefährlichen Viren, Orte mit radioaktiver Strahlung oder in Katastrophengebieten, ist die Fernsteuerung des Roboters sinnvoll.

Ein weiteres Anwendungsgebiet wäre die Reisebranche. Kunden eines Reisebüros könnten sich mit einer Vorabbesichtigung, durch den Teleroboter, über einen Reiseort besser überzeugen als dies mittels einfachen Katalogfotos möglich wäre.

Head-mounted Displays, die dem Anwender das Abbild der entfernten Umgebung präsentieren, gibt es viele auf dem Markt. Sie sind aus der Spieleindustrie als VR-Brillen bekannt. Die Funktionalität, die solch ein Produkt bietet, kann auch durch ein herkömmliches Smartphone realisiert werden. Mit einem preiswerten Gehäuse kann ein Smartphone in eine VR-Brille verwandelt werden. Dieser Ansatz ist wesentlich kostengünstiger für den Anwender, da er das Smartphone bereits mitbringt.

Ein Telepräsenzsystem erweitert die Sinneswahrnehmung des Anwenders um die Eindrücke der entfernten Umgebung. Die Technologie der Augmented Reality (AR) beschreibt die Erweiterung der Realitätswahrnehmung des Anwenders. Das im Rah-

men dieser Arbeit entwickelte System kann demnach der Augmented Reality untergeordnet und als ein AR-System betrachtet werden.

1.2 Zielsetzung und Anforderungen

Das Ziel dieser Arbeit ist die Entwicklung eines Telepräsenzsystems zur Auslagerung der visuellen Wahrnehmung des Benutzers an einen entfernten Ort. Der daraus entstehende Prototyp soll Aufschluss darüber geben, ob es möglich ist, solch ein System zu entwickeln.

Dieses System soll hierbei folgende Anforderungen erfüllen:

- Aufbau eines Kameraroboters mit drei Freiheitsgraden
- Implementierung der Software zur:
    - Erfassung der Umgebung durch eine stereoskopische Kamera.
    - Übertragung und Visualisierung bewegter Bilder auf einem Head-mounted Display.
    - Synchronisierung der Kamerabewegung des Roboters mittels der Kopfbewegung des Anwenders.

Ein Telepräsenzsystem stellt hohe Anforderungen bezüglich der Echtzeitfähigkeit. Die Übertragung der Videodaten und der Kopfbewegung des Anwenders, soll mit einer geringen Latenzzeit erfolgen, um dem Anwender eine flüssige und natürliche Wahrnehmung der entfernten Umgebung zu ermöglichen (mindestens 30 Einzelbilder und Positionsänderungen der Kameraausrichtung pro Sekunde).

Die Hardware, die in dieser Arbeit den Aufbau des Kameraroboters realisieren soll, muss diesen Anforderungen gerecht werden.

Als Head-mounted Display soll ein Android-Smartphone zum Einsatz kommen, welches folgende Kriterien an die Software (App) stellt:

- Visualisierung der vom Kameraroboters gesendeten bewegten Bilder
- Erfassung der Kopfbewegung des Anwenders und der Übertragung dieser an den Kameraroboter

Die Anwendung soll von mehreren Anwendern simultan benutzt werden können, wobei immer nur ein Anwender die Kontrolle des Kameraroboters übernehmen kann.

Am Ende der Arbeit werden die Ergebnisse vorgestellt (Kapitel 5 - Validierung).

1.3 Telepräsenzsystem

Die Telepräsenz ist eine Technologie, die es einer Person, also dem Anwender eines Telepräsenzsystems, ermöglicht, durch ein Kommunikationsmedium an einem entfernten Ort einzutauchen und sich dort anwesend zu fühlen. Der Anwender hat das Gefühl, in der anderen Umgebung zu sein, in der er nicht physisch anwesend ist. Dabei kommen Systeme aus der Telerobotik zum Einsatz. Die Telerobotik beschäftigt sich mit der Fernsteuerung von Robotern. Ein Teleroboter vertritt dabei den Anwender in der entfernten Umgebung.

Als Immersion wird die Eigenschaft beschrieben, die das Gefühl vermittelt, in einer künstlichen oder realen, aber entfernten Umgebung einzutauchen. Künstliche bzw. computergenerierte Umgebungen sind aus dem Kontext der *Virtual Reality* bekannt (siehe 2.3). Die computergestützte Erweiterung der Realitätswahrnehmung wird als *Augmented Reality* bezeichnet (siehe 2.1).

Je höher der Immersionsgrad ist, desto mehr fühlt sich der Anwender in die entfernte Umgebung versetzt. Der Grad der Immersion ist dabei abhängig von mehreren Faktoren.

Die Qualität der Darstellung spielt eine essentielle Rolle. Sieht der Anwender ein hochaufgelöstes Bild der Umgebung und kann räumliche Tiefe wahrnehmen, steigt das Gefühl, sich vor Ort zu befinden. Der visuelle Eindruck stellt für den Menschen zwar den wichtigsten Sinneseindruck dar, jedoch tragen auch alle anderen Sinneseindrücke (z.B. Gehör- und Tastsinn) dazu bei, den Immersionsgrad zu steigern.

Zusätzlich beeinflussen Interaktionen des Anwenders den Immersionsgrad. Je weniger die Benutzerschnittstelle zur Steuerung des Teleroboters als „künstlich“ wahrgenommen wird, desto mehr hat der Anwender das Gefühl, in der entfernten Umgebung zu interagieren.

Dieses Präsenzgefühl lässt sich nicht objektiv messen und ist davon abhängig, wie realitätsnah die Umgebung von einer individuellen Person wahrgenommen wird [Slater 2003].

Ein Telepräsenz-System verfolgt, je nach Anwendungsbedarf, das Ziel, so viele Sinneseindrücke aus der entfernten Umgebung wie möglich zu dem Anwender des Systems, auch Teleoperator genannt, zu transportieren. Wie gut ein System hinsichtlich dieser Eigenschaft funktioniert, hängt von den Benutzerschnittstellen zum Teleroboter ab.

Abhängig von den zur Verfügung gestellten Benutzerschnittstellen wird die Telepräsenz von der Teleoperation abgegrenzt. Abbildung 1 und 2 verdeutlichen den Übergang von der Teleoperation zu der Telepräsenz.

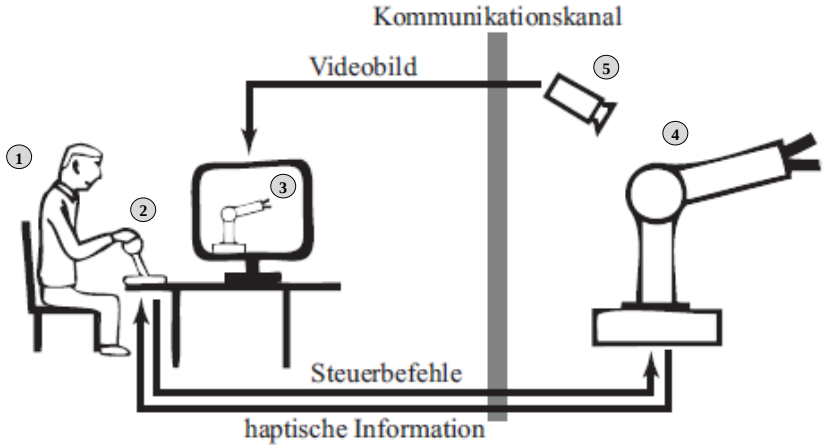


Abbildung 1: Teleoperations-System (Abgeänderte Version von [Rößler 2009])

Unter der Teleoperation (auch Telemanipulation genannt) versteht man die Ausführung von Aufgaben mit Hilfe eines stationären oder beweglichen Roboters in einer entfernten Umgebung. Abbildung 1 zeigt den schematischen Aufbau eines Systems zur Teleoperation. Dabei wird der Roboter (4) von einem stationären Benutzer gesteuert (1). Die entfernte Umgebung wir über eine Video-Kamera überwacht (5). Der Benutzer sieht dieses Bild auf einem Monitor und kann das Geschehen überwachen (3). Die Benutzerschnittstelle zur Steuerung des Roboters ist meist ein Joystick oder ein ähnliches Eingabegerät mit Kraftrückkopplung (2). Solch eine Benutzerschnittstelle stellt für den Benutzer einen hohen Abstraktionsgrad dar und lässt sich oft nicht intuitiv steuern. Der Benutzer braucht eine hohe Aufmerksamkeit bei der Bedienung und es bedarf eine Eingewöhnungszeit in die Steuerung. Die Aufmerksamkeit fehlt dem Benutzer bei der Ausführung der wesentlichen Aufgaben mit dem Roboter.

Im Gegensatz zur Teleoperation liegt bei der Telepräsenz nicht das Agieren in der entfernten Umgebung im Vordergrund, sondern das Gefühl der Präsenz, also die Wahrnehmung der entfernten Umgebung. Der Benutzer taucht dabei mit so vielen seiner Sinneswahrnehmungen wie möglich in die entfernte Umgebung ein. Dabei helfen Technologien aus der Virtual Reality.

Der Vorteil der Telepräsenz liegt darin, dass der Benutzer seine volle Aufmerksamkeit auf die eigentlichen Aufgaben legen kann, da ihm eine intuitive und realitätsnahe Steuerung und Wahrnehmung zur Verfügung stehen. Das Ziel ist es, die Steuerung und Wahrnehmung vollkommen transparent für den Benutzer zu gestalten, um so einen möglichst hohen Immersionsgrad zu erzeugen.

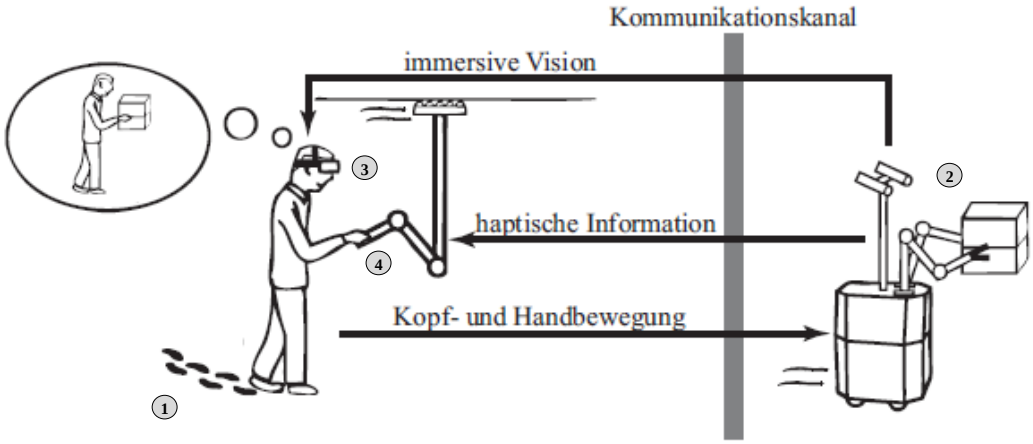


Abbildung 2: Telepräsenz-System (Abgeänderte Version von [Rößler 2009])

Eine Steigerung der einfachen Telepräsenz ist die weiträumige Telepräsenz. Dabei ist der Benutzer nicht mehr stationär an einen Ort zur Bedienung des Systems gebunden. Nicht nur die Kopfbewegungen des Benutzers werden aufgezeichnet und in die entfernte Umgebung übersetzt, auch seine freie Bewegung wird übertragen. Hierbei können mobile radbasierte Roboter eingesetzt werden.

Abbildung 2 zeigt den Aufbau eines Telepräsenz-Systems. Die Bewegung des frei beweglichen Benutzers kann aufgezeichnet werden (1), der mobile Teleroboter setzt diese Bewegung, in der entfernten Umgebung um (2). Der Benutzer sieht das Abbild der entfernten Umgebung nicht mehr über einen klassischen Monitor, er trägt eine Display-Brille (Head-mounted Display, siehe 2.4) zur Darstellung der entfernten Umgebung (3). Somit werden die visuellen Sinne des Benutzers von seiner realen Umgebung ausgeblendet. Über eine mobile haptische Schnittstelle (Handschuhe mit Kraftsensor oder ein Manipulatorarm wie in Abbildung 2 dargestellt) kann der Benutzer in der entfernten Umgebung agieren (4). Hierbei können die Handbewegungen und Aktionen des Benutzers aufgezeichnet und z.B. durch einen Greifarm des Teleroboter umgesetzt werden.

Es können verschiedene Teilmengen der oben beschriebenen Eigenschaften in einem Telepräsenzsystem vorkommen. Je nach Anwendungsfall muss das System z.B. keine weiträumige Bewegung oder haptische Schnittstellen aufweisen. In dieser Arbeit soll ein System zur reinen visuellen Immersion entwickelt werden.

Unter den Begriff Telepräsenz fallen auch einfache Systeme, die bei Meetings zur klassischen Videokonferenz verwendet werden, sobald eine sehr hohe Auflösung und eine lebensgroße Darstellung der Teilnehmer bei der Videotelefonie vermittelt werden.

[Jehle 2014] definiert diese Technologie der Telepräsenz mit dem Begriff *„Moved Reality“*. Es handelt sich dabei um eine reale, aber verschobene Realität. Der Grad der Wahrnehmung kann laut [Jehle 2014] in sechs Stufen eingeteilt werden:

- Visuelle Wahrnehmung
- Auditive Wahrnehmung
- Olfaktorische Wahrnehmung (Geruch)
- Gustatorische Wahrnehmung (Geschmack)
- Taktile Wahrnehmung (Gefühl)
- Vestibuläre Wahrnehmung (Gleichgewicht)

Das Erreichen eines Immersionsgrades der Stufe 6 würde eine vollständige Auslagerung des menschlichen Bewusstseins auf eine Maschine bedeuten. Dies stellt das „ultimative“ Ziel der Forschung dar, ist jedoch nach heutigem Stand der Technik nicht realisierbar. Es gibt allerdings bereits einige Forschungsprojekte, die diesem Ziel näher kommen sollen.

Der Tast- und Gefühlssinn könnte von einem mit einer hochauflösenden Sensorhaut[1] [Wu 2013] ausgestatteten Roboter an eine pneumatische Weste, wie sie das Unternehmen TN Games für die Spieleindustrie entwickelt hat, an den Benutzer übertragen werden. Einen weiteren Ansatz bilden Force-Feedback-Handschuhe, mit denen eine künstliche Tastwahrnehmung realisiert werden kann. Diese Geräte werden auch als haptische Schnittstelle bezeichnet.

Um Gerüche zu übertragen, könnten Technologien aus der *Scentography* zum Einsatz kommen. Ein Gerät zur Erstellung von Gerüchen, basierend aus 128 Basis-Stoffen, wurde bereits von der Firma DigiScents entwickelt. Zur Aufzeichnung von

[1] http://www.spektrum.de/news/eine-hochaufloesende-sensorhaut-fuer-roboter/1192468

Gerüchen gibt es bereits Ansätze zur Zerlegung des Geruchs in einzelne Basisgerüche. Das Übertragen von Geschmäckern ist theoretisch nach dem gleichen Prinzip möglich.

Ein sogenannter „Virtual-Reality-Laufstall“ aus der Spielebranche mit dem Namen *Omni*[2] könnte das Gefühl des Laufens und des Gleichgewichtsinns realistischer vermitteln.

Lediglich eine Annäherung an die 6. Stufe der Sinnesauslagerung ist somit möglich. Was der Sinn und Anwendungszweck solch eines Telerobotik-Prototyps ist, bleibt dabei offen.

In dieser Arbeit wird Stufe Eins erreicht, da nur die visuelle Wahrnehmung auf eine Maschine bzw. ein Robotersystem ausgelagert wird.

Auf Hardwareebene der Anwenderseite ist die Telepräsenz üblicherweise eine Variante der Virtual Reality, da dieselbe Ausrüstung benötigt wird, um die Bewegung des Anwenders aufzuzeichnen und das „verschobene“ Abbild der Realität für den Anwender zu visualisieren.

[2] http://www.virtuix.com

2. Stand der Technik

Dieses Kapitel gibt einen Überblick über bereits existierende Lösungen, für die in dieser Abschlussarbeit erläuterten Ziele. Zusätzlich werden die Grundlagen der verwendeten Technologien zum Verständnis des Themas beschrieben. Die Grundlagen sind nicht nur für die Telepräsenz relevant. Sie kommen auch in anderen Bereichen zur Anwendung.

2.1 Augmented Reality

Die Augmented Reality (AR), also die „erweiterte Realität“, ist die computergestützte Abbildung der Realität, welche mit künstlichen Informationen oder Inhalten erweitert wird. Die am häufigsten zitierte Definition in der Literatur ist von [Azuma 1997] und beschreibt die visuelle Augmented Reality mit folgenden Eigenschaften:

- Kombination aus der Realität und Virtualität mit einem 3-dimensionalen Bezug zueinander.
- Interaktiv und in Echtzeit

In den meisten Anwendungsfällen wird die Augmented Reality als eine rein visuelle Erweiterung von Bildern und bewegten Bildern verstanden. Das Abbild der Realität wird hierbei durch das Einblenden von Informationen (Overlay) angereichert oder mit virtuellen Objekten ausgestattet, als wären sie Gegenstände in der Realität. Am häufigsten treffen wir bei TV-Übertragungen von Sportveranstaltungen auf den konkreten Einsatz der Augmented Reality. Bei einem Fußballspiel wird in einer Freistoßsituation die Entfernung zum Tor oder die virtuelle Abseitslinie in der Wiederholung eingeblendet. Ein weiteres Beispiel der reinen Anreicherung der Realität durch Zusatzinformationen ist eine Anwendung auf einem Smartphone, welche für den Anwender nützliche Zusatzinformationen zu einem bekannten Gebäude oder Ort einblendet (Abbildung 3).

Abbildung 3: Augmented Reality App Wikitude

In der Luftfahrt werden die Piloten bei Nacht und schlechten Sichtverhältnissen mit dem Instrumentenlandesystem (ILS) grafisch mit virtuellen Linien der Landebahn und Informationen zur Sinkrate auf dem Head-Up-Display informiert.

Die Augmented Reality ist älter als gedacht, bereits Mitte der 60er Jahre wurde eine der ersten AR-Anwendungen entwickelt. Unter dem Namen *Sword of Damocles* veröffentlichte Ivan Sutherland seine Forschungsergebnisse über ein System, welches die Orientierung einer tragbaren Display-Brille benutzte und abhängig von der Kopfbewegung, ein Bild aus einem Drahtgittermodell erstellte. Das generierte Bild wurde zusammen mit der Realität auf der Brille dargestellt. Das Projekt wurde - wie so vieles - durch das Militär finanziert.

Die Augmented Reality ist nicht ausschließlich auf die visuelle Darstellung beschränkt. So könnten theoretisch alle realen Sinneswahrnehmungen mit künstlichen Eindrücken erweitert werden. Welche AR-Anwendung realisierbar ist, hängt davon ab, ob die dafür benötigte Technologie (Aufnahme und Wiedergabe der Sinneseindrücke) existiert. Da für den Menschen der Sehsinn und das Gehör die wichtigsten Wahrnehmungssinne zur Aufnahme von Informationen darstellen, sind die meisten AR-Anwendungen auch auf diese beiden Sinne ausgelegt. Andere Sinneseindrücke (Geruch-, Geschmack und Tastsinn) können technisch nur schwer bis gar nicht künstlich erweitert werden. Lediglich Force-Feedback-Geräte wie Joysticks oder Lenkräder und ganze Apparaturen, wie sie z.B. bei professionellen Flugsimulatoren vorkommen, machen es möglich, den Tast- und Gleichgewichtssinn mit virtuellen Inhalten bis zu einem gewissen Grad anzureichern.

Es gibt heutzutage eine Vielzahl an praktischen Anwendungsgebieten für die Augmented Reality. Zusätzlich zu dem oben beschriebenen Einsatz in der zivilen Luftfahrt hat das Militär ein großes Interesse an der AR. In der Medizin und der Industrie kann die AR bei Operationen bzw. Montagearbeiten nützlich sein. Einen didaktischen Einsatz in der Lehre verknüpft Arbeitsblätter und Lehrbücher mit virtuellen Lerninhalten. Ein Beispiel dazu ist die App *„Anatomy 4D“* für Android und iOS Geräte, die Arbeitsblätter mit anschaulichen virtuellen und dreidimensionalen Objekten des menschlichen Körpers erweitert und dem Benutzer das Lernen auf eine attraktive und spannende Weise erfahren lässt. In der Unterhaltungsindustrie revolutioniert die Augmented Reality die Spielebranche und unterstützt den Zuschauer bei Sportübertragungen mit Zusatzinformationen. Spielekonsolen wie die Playstation von Sony setzen vermehrt auf die AR. Auch Werbekampagnen im Marketing und die

Navigation sind Beispiele dafür, in welchen Bereichen die AR heutzutage zum Einsatz kommt.

2.2 AR-System

Ein System, das die technischen Komponenten vereint, die es benötigt, eine AR-Anwendung aufzubauen, wird als Augmented Reality System (kurz „AR-System“) bezeichnet. Dabei kommen häufig folgende technische Komponenten zum Einsatz: Kamera, Trackinggerät, AR-Display und das Software-System.

Ein Kamerasystem nimmt die reale Szene auf. Nach einer Beschreibung von [Koller u. a. 1997, S. 1] heißt es: *„A video-based AR system essentially has two cameras, a real one which generates video of the real environment, and a virtual one, which gen-erates the 3D graphics to be merged with the live video stream. Both cameras must have the same internal and external parameters in order for the real and virtual objects to be properly aligned.”*

Die zwei Bildquellen müssen in einem räumlichen Zusammenhang verknüpft werden. Um die räumliche Lage des AR-Systems bzw. des Ausgabegerätes zu ermitteln, werden Tracking-Technologien eingesetzt. Die Lageinformationen können mittels GPS und Sensoren wie einem Beschleunigungssensor (Accelerometer) und dem Gyroskop ermittelt werden.

Eine Anwendung (Software) ist für die Logik und die Inhalte zuständig. Die Lage und perspektivische Abbildung der virtuellen Objekte wird aus den Trackingdaten, also der Position und Blickrichtung des Kamerasystems bzw. des Anwenders, berechnet.

Die Darstellung des fertigen Bildes wird dem Benutzer durch ein Display zur Verfügung gestellt. Hierbei können, je nach AR-Anwendung, verschiedene Display-Technologien zum Einsatz kommen. Die am häufigsten verwendeten Ausgabegeräte können in vier Kategorien eingeordnet werden: herkömmliche Displays, Head-up Displays, Projektoren und Head-mounted Displays.

Bei vielen einfachen AR-Anwendungen kommen Monitore, Tablet-PCs oder Smartphones als Ausgabegerät zum Einsatz. Die zunehmende Verbreitung von mobilen Endgeräten haben diese zu einem massentauglichen Medium für Augmented Reality Systeme heranwachsen lassen.

Head-up Displays funktionieren nach dem *optical see-through-Prinzip* und finden Anwendung im Cockpit von Flug- und Fahrzeugen. Dabei wird das computergenerierte Bild auf einer sichtdurchlässigen Projektionsfläche dargestellt.

Bei der Verwendung von Projektoren als AR-Ausgabegerät können virtuelle Bilder direkt auf die Oberfläche in der realen Umgebung projiziert werden. Der Vorteil darin liegt in der Lageunabhängigkeit des Anwenders, da die virtuellen Inhalte nicht abhängig von der Position des Betrachters neu berechnet werden müssen und mehrere Anwender die gleichen Inhalte sehen können. Für die meisten AR-Anwendungen ist diese interessante Variante jedoch nicht geeignet (Abhängigkeit zum Raum bzw. Projektionsfläche).

Das Head-mounted Display (HMD) ist ein Ausgabegerät, welches der Benutzer auf seinem Kopf trägt und das Bild auf einem Display darstellt oder direkt auf die Netzhaut des Auges projiziert. Siehe Kapitel 2.4.

Ein populäres Beispiel eines AR-Systems ist das Google Glass Projekt. Hierbei wird ein Head-up-Display zur Darstellung von Informationen eingesetzt.

Im Rahmen dieser Abschlussarbeit wird ein AR-System mittels eines Smartphones als Head-mounted Display entwickelt.

2.3 Virtual Reality

Im Vergleich zur Augmented Reality ist die Virtual Reality (VR), die „virtuelle Realität“, eine rein computersimulierte Umgebung, also eine künstliche computergenerierte Welt, die neben der Realität existiert. Der Benutzer einer Virtual Reality-Anwendung taucht in diese virtuelle Umgebung ein. Dabei beschränken sich die Anwendungen meist auf visuelle und auditive Sinneseindrücke. Wie bei der Augmented Reality kann die VR auch virtuelle Tast-, Geruchs- und Geschmackseindrücke einschließen.

Abbildung 4: Virtual Reality Erlebnis mit der Oculus Rift

Die Virtual Reality lässt sich in drei Schwerpunkte einteilen:

- *Visualisierung* des virtuellen Inhaltes
- *Immersionsgrad* bzw. Grad des Nachempfindens der virtuellen Umgebung.
- *Interaktivität* mit der virtuellen Umgebung.

Dem Benutzer wird die virtuelle Umgebung über Displays, die um ihn herum aufgebaut sind, oder mittels Head-mounted Displays präsentiert.

Ein hauptsächliches Anwendungsgebiet der VR ist die Computerspieleindustrie sowie Flug- und Fahrsimulatoren.

Auch im Hinblick von Lehrinhalten wird der Einsatz der virtuellen Realität diskutiert [Forbes]. Ihr Einsatz soll dafür sorgen, dass Schüler Erfahrungen sammeln können, die in ihrer Realität nicht oder nur schwer möglich wären. So könnten DNA-Stränge in der Biologie oder altertümliche Gebäude und Ereignisse der Geschichte virtuell betrachtet werden. Der Bereich des E-Learning kann durch die VR neue Inhalte gewinnen. *„Doch während Gestaltungsanforderungen für klassische Medien gut erforscht sind, liegen über virtuelle Welten bisher nur wenige Erkenntnisse aus lernpsychologischer und gestalterischer Perspektive vor.“* [Schwan u. Buder 2006, S. 1].

Das Phänomen, dass künstliche Welten neben der echten Welt existieren, hat es auch schon vor der Einführung des Computers gegeben. Beispiele sind Märchen und ande-

re Geschichten. Auch in der Malerei lassen sich frühe Formen der „virtuellen Welt“ erkennen.

2.4 Head Mounted Display

Bei Virtual Reality-Systemen kommt teilweise die gleiche Technologie wie bei AR-Systemen zum Einsatz. Lediglich der Zweck beider Systeme ist ein anderer. Der Anwender eines VR-Systems ist von seiner realen Umgebung völlig abgeschottet und nimmt nur virtuelle Sinneseindrücke wahr.

Zur Darstellung der reinen virtuellen und der erweiterten Realität, also für den Transport des Abbildes zum Anwender, existieren verschiedene Arten von Ausgabegeräten, sei es das Smartphone-Display, das die Informationen zur Freiheitsstatue in einer Kamera-Anwendung einblendet, das Head-Up-Display des Piloten oder eine sogenannte Datenbrille, auch Head-mounted Display genannt.

Ein Head-mounted Display (HMD) ist ein vom Anwender auf dem Kopf getragenes optisches Ausgabegerät, das folgende zwei Aufgaben erfüllt: Tracking und Visualisierung. Es ist dafür zuständig, die Kopfbewegung des Benutzers aufzuzeichnen und ihm ein digitales Bild zu präsentieren. Das HMD ist meist mit einem Rechner verbunden, der die Trackinginformationen verarbeitet und abhängig davon das Bild generiert. Abbildung 4 (S. 12) zeigt ein HMD im Einsatz.

2.5 Das Stereoskopische Bild

Ein HMD besteht aus zwei separaten zweidimensionalen Displays. Damit der Benutzer eine visuelle und dreidimensionale Wahrnehmung erhält, bekommt jedes Auge ein perspektivisch leicht unterschiedliches Bild präsentiert. Dabei wird ein räumlicher Eindruck von Tiefe erzeugt, obwohl es sich um zwei 2D-Bilder handelt (Abbildung 5). Man spricht dabei von einem stereoskopischen Bild. Das menschliche Sehen beruht auf dem gleichen Prinzip. Das rechte und das linke Auge nehmen gleichzeitig ein Bild aus zwei unterschiedlichen Blickwinkeln auf. Das Gehirn kann dadurch die Entfernung der betrachteten Objekte im Bild erstellen und so ein räumliches Bild des Sichtfeldes erzeugen.

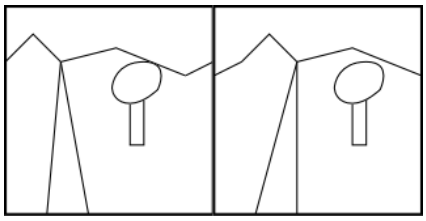


Abbildung 5: Zwei Bildteile eines stereoskopischen Bildes

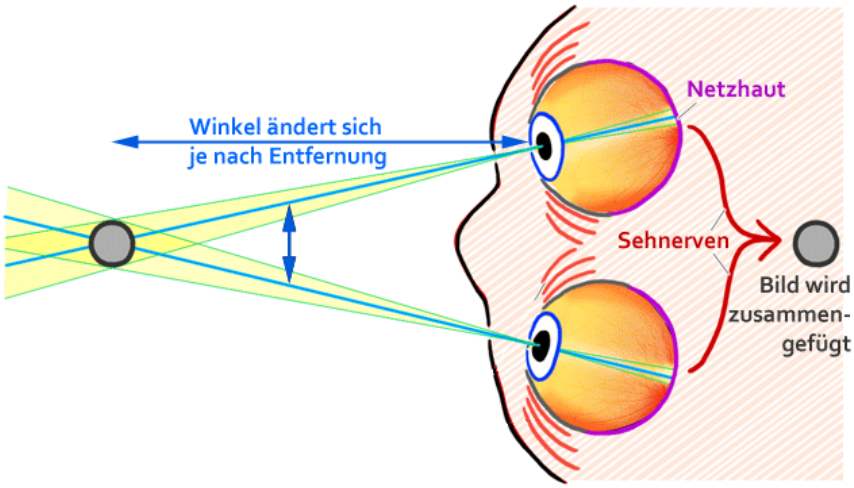


Abbildung 6: Stereoskopisches Sehen (Quelle: sehtestbilder.de)

Durch den unterschiedlichen Blickwinkel und dem Abstand beider Augen werden Objekte, die in der Nähe des Betrachters liegen, mit dem linken Auge etwas mehr von der einen Seite und mit dem rechten Auge mehr von der anderen Seite gesehen. Je weiter das betrachtete Objekt vom Betrachter entfernt liegt, desto kleiner wird der Winkel zwischen beiden Augen (blaue Sichtlinien in Abbildung 6). Bei einem weit entfernten Betrachtungspunkt nähern sich die Sichtlinien einer parallelen Ausrichtung an. Aus diesen und weiteren Informationen erzeugt das Gehirn die räumliche Wahrnehmung.

Für ein künstlich erstelltes stereoskopisches Bild müssen zwei Kameras beide Teilbilder synchron aufnehmen. Im Falle der virtuellen Realität handelt es sich dabei um virtuelle Kameras, also zwei virtuelle Blickwinkel in der computergenerierten Umgebung. Beide Kameras müssen einen seitlichen Abstand von 65 mm zueinander aufweisen. Dieser Abstand entspricht einem mittleren Augenabstand des Menschen [Dodgson 2004]. Damit ein bestmöglicher räumlicher Eindruck entsteht, muss der Betrachter beide Teilbilder mit einem Augenabstand betrachten, der der Brennweite bei der Aufnahme gleicht. Zusätzlich darf sich der Sehwinkel nicht zur Aufnahme unterscheiden. VR-Brillen realisieren dies nach diesem Prinzip. Ein angeschlossener und leistungsstarker Rechner generiert das stereoskopische Bild. Die VR-Brille visualisiert dabei ein Teilbild pro Auge.

2.6 Oculus Rift

Der bekannteste Vertreter unter den Head-mounted Displays ist die Oculus Rift der Firma Oculus VR. Finanziert wurde dieses Projekt auf der Crowdfunding-Plattform Kickstarter und erreichte im Jahr 2012 Spenden von insgesamt 2,4 Millionen US-Dollar. Das Finanzierungsziel von 250.000 US-Dollar wurde in kürzester Zeit erreicht. Das zeigt die große Akzeptanz der breiten Masse der Virtual Reality. Auch der Internetkonzern Facebook erkannte das Potential und übernahm im März 2014 das Unternehmen Oculus VR für zwei Milliarden US-Dollar.

Die Oculus Rift steht für Entwickler zum jetzigen Stand als Developer Kit der Version Zwei (DK2) mit einer Auflösung von *960 x 1080* Pixeln pro Auge und einem Sichtfeld von 100° zur Verfügung. Die erste Version für Endkunden soll Ende 2015 veröffentlicht werden. Das Oculus Rift DK2 gibt es bereits für 350 US-Dollar.

Eine Erweiterung der ersten Version (DK1) stellt nicht nur ein verbessertes Display mit höherer Auflösung dar, zusätzlich wurde die Head-Tracking Komponente verbessert. Zur Bestimmung der Kopflage wird auf Infrarot-LEDs gesetzt, die mit einer Kamera aufgezeichnet werden.

Neben der Oculus Rift, existieren noch weitere Produkte auf dem Markt. Sony entwickelt derzeit eine VR-Brille. Unter dem Namen *Morpheus* steht sie für die Playstation-Spielekonsole zur Verfügung.

2.7 Smartphone als Head-mounted Display

Aus einem gängigen Smartphone und einer dafür vorgesehenen Halterung kann eine vollwertige VR-Brille erstellt werden. Möglich macht dies die Technik eines heutigen Smartphones. Sensoren zur Lagebestimmung des Gerätes (2.12), die sonst hauptsächlich als Steuerung für Spiele eingesetzt werden, ermitteln die Kopfbewegung des Benutzers. Eine App verarbeitet die Kopfbewegung und stellt ein Bild je Auge auf dem smartphone-eigenen Display in zwei Hälften geteilt dar (Abbildung 7). Zwei verstellbare Linsen vergrößern die Darstellung nach dem gleichen Prinzip, wie es bei den herkömmlichen Head-mounted Displays der Fall ist.

Abbildung 7: Smartphone als VR-Brille

Dieser Ansatz, bei dem das Smartphone als Recheneinheit und Ausgabegerät herhält, bringt einige Vorteile mit sich. Das Brillengestell kann kostengünstig produziert werden. Zusätzlich bringt der Benutzer in der Regel das Smartphone bereits mit. Wie eine Studie der Internet-Marktforschungsfirma Comscore zeigt, wächst der Absatz von modernen Smartphones stätig [ComScore 2014]. Jeder zweite deutsche besitzt ein Smartphone (laut [ComScore 2014] sind dies 41,1 Mio. im Jahr 2014).

Das Unternehmen Shoogee GmbH & Co KG vertreibt solch eine Halterung mit dem Namen Durovis Dive (Abbildung 7, rechts). Während die meisten VR-Brillen wie Sonys Morpheus und die Oculus Rift nicht unter mehreren Hundert Euro zu haben sind, ist die Durovis Dive mit ca. 50 Euro verhältnismäßig billig. Allerdings darf bei der Preisfrage nicht vergessen werden, dass ein modernes Smartphone benötigt wird, falls dies nicht zur Verfügung steht.

Das Smartphone muss über einen Beschleunigungs- und Magnetfeldsensor verfügen, um das Head-Tracking korrekt zu realisieren. Da die Anzeige für beide Augen geteilt wird, darf das Display keine zu geringe Display-Auflösung vorweisen. Eine zu geringe Auflösung führt zu einem sogenannte „Fliegengittereffekt“, der bereits von der

ersten Entwicklerversion der Oculus Rift bekannt ist. Aufgrund des geringen Betrachtungsabstands zum Display beim Tragen der Brillenhalterung sind die einzelnen Pixel erkennbar, was die Immersion deutlich verschlechtert. Ein weiteres entscheidendes Element ist die Bildwiederholungsrate des Smartphone-Displays. Für schnelle Reaktionszeiten bei bewegten Bildern sollte diese nicht zu gering ausfallen.

Für die vorliegende Arbeit wurde ein Nexus 5 mit einer Display-Auflösung von *1920 x 1080* Pixeln (Full-HD) verwendet. Das entspricht einer effektiven Auflösung von *960 x 1080* Pixeln pro Auge. Mit dieser Display-Spezifikation kann somit einem Auge kein hochauflösendes Bild in Full-HD präsentiert werden. Die erste Entwicklerversion der Oculus Rift hat im Vergleich dazu eine Auflösung von nur *640×800* Pixeln je Auge. Ab der zweiten Entwicklerversion verfügt die Oculus Rift über eine Auflösung von *960 x 1080* Pixeln je Auge, was der Darstellung des Nexus 5 gleich kommt.

Tabelle 1 (S. 18) zeigt den Unterschied zwischen den Technologien.

VR-Brillen wie die Oculus Rift benötigen einen PC als Recheneinheit. Dient das Smartphone als VR-Brille, werden alle Rechenleistungen vom Smartphone erbracht. Das ist zudem platzsparend, hat jedoch den Nachteil der vergleichsweise geringeren Rechenleistung eines Smartphones. Beispielsweise ist bei der Oculus Rift die grafische Aufbereitung durch den PC mit dem Einsatz von leistungsstarken Grafikkarten wesentlich performanter. Abhängig von der Anwendung kann das Smartphone mit der klassischen VR-Brille nicht mithalten. Da in dieser Arbeit das mobile Gerät als eine Art „Thin-Client“ agiert - lediglich die simple Darstellung, das Head-Tracking und die Kommunikation fallen in die Verantwortung des Gerätes - kann dieser Nachteil vernachlässigt werden.

Produkt	Display	Gewicht (Gramm)	Preis
Oculus Rift DK 1	1280x800	369	300 €
Oculus Rift DK 2	1920 x 1080	440	326 €
Durovis Dive (mit Nexus 5)	1920 x 1080	150	50 €
Google Cardboard (mit Nexus 5)	1920 x 1080	Gewicht des Smartphones	ca. 15 € (Amazon)
Sony Morpheus	1920x1080	keine Angabe	Release: 2015

Tabelle 1: Vergleich von HMD Technologien

2.8 Google-Cardboard

In dieser Arbeit kommt das Google-Cardboard als AR-Display zum Einsatz. Dabei handelt es sich um eine vom Unternehmen Google entwickelte „Do-it-yourself"-VR-Brille zum Selberbauen. Die Halterung des Smartphones ist aus Karton, beinhaltet zwei Linsen und ermöglicht es, das Smartphone mit einem Klettverschluss zu fixieren. Aus einem Smartphone und dem Bausatz kann dadurch in wenigen Handgriffen eine vollwertige und funktionstüchtige VR-Brille gebastelt werden.

Das Cardboard wurde auf der I/O, einer Entwicklerkonferenz von Google, vorgestellt und als Bausatz der Öffentlichkeit zur Verfügung gestellt. Der Bausatz kann für ca. 15€[3] im Internet bestellt werden.

Außerdem befindet sich ein Magnet in der Halterung, über den sich die ausgeführte Anwendung steuern lässt. Dabei reagiert die Anwendung auf die Veränderung des Magnetfeld-Sensors. Der Benutzer hat somit die Möglichkeit auf simple Interaktionen mit der Anwendung. Das ist hilfreich, da für alle anderen Interaktionen das Touch-Display von der Halterung des Cardboards verdeckt wird.

Abbildung 8: Google Cardboard - Head-mounted Display aus Karton und einem Smartphone.

Im Gegensatz zu den anderen Halterungen, die dazu dienen, ein Smartphone in eine VR-Brille zu „verwandeln", ist das Cardboard wesentlich kostengünstiger. Der schwedische Autohersteller Volvo hat das Cardboard als Marketing Instrument erkannt und lädt auf eine virtuelle Testfahrt in einem neuen Produkt ein.

Google stellt für Entwickler, die Anwendungen für das Google-Cardboard erstellen, ein Software Development Kit (SDK) als Open Source Software zur Verfügung. Das SDK bietet eine Schnittstelle für die oben genannten Funktionalitäten. Die Visualisierung erfolgt mit OpenGL für Android oder der Unity Game Engine. Zum Ausglei-

[3] http://www.amazon.de/AM-CARDBOARD%C2%AE-Complete-Cardboard-Project/dp/B00LM36DUK

chen der Verzerrungen der optischen Linsen bietet das SDK einen OpenGL Shader-Filter an.

2.9 Android

Das „Mobile Computing“ Labor der Hochschule Offenburg beschäftigt sich hauptsächlich mit der Entwicklung von mobilen Anwendungen auf Basis von Android.

Android ist ein quelloffenes und freies Betriebssystem für mobile Endgeräte wie Smartphones und Tablet-PCs, das von Google und der Open Handset Alliance entwickelt wurde. Android wurde im Oktober 2008 erstmals veröffentlicht [Google02]. Android hat einen großen Marktanteil im mobilen Sektor und verzeichnet täglich 1,5 Millionen Aktivierungen von neuen Android-Geräten.

Laut dem Marktforschungsunternehmen Gartner hat das Android-Betriebssystem einen Marktanteil von 80,7 % im Jahr 2014 [Gartner 2015]. Tabelle 2 zeigt die Entwicklung der letzten Jahre.

	2011	2012	2013	2014
Android	46,66 %	66,4 %	78,5 %	80,7 %
iOS	18,88 %	19,1 %	15,5 %	15,4 %
Microsoft	1,85 %	2,5 %	3,2 %	2,8 %
BlackBerry	10,9 %	5 %	1,9 %	0,6 %
Andere	21,71 %	7 %	0,9 %	0,5 %

Tabelle 2: Marktanteile der Betriebssysteme am weltweiten Endkundenabsatz [Gartner 2015]

Android bietet eine standardisierte Plattform zur Entwicklung von Anwendungen und garantiert die geräteunabhängige Ausführung. Ist die Anwendung kompatibel zum verwendeten Gerät, so kann sie über einen Marktplatz für Anwendungen (Google Play) ausgeliefert werden.

Die Entwicklung von Anwendungen für Android erfolgt mit Hilfe des von Google zur Verfügung gestellten Software Development Kits (Android SDK) in der Programmiersprache Java.

Ein Rechtemanagementsystem sorgt dafür, dass Anwendungen nur so viele Rechte bekommen, wie sie für den Betrieb benötigen. Um zusätzliche Sicherheit zu gewährleisten, läuft eine Anwendung in einer isolierten und geschützten Umgebung (Sandbox).

2.10 Sensoren in Android

Moderne Geräte, auf denen Android läuft, sind mit einer Vielzahl von Sensoren ausgestattet. Ein Sensor ist ein technisches Bauteil, mit dem das Gerät verschiedene Informationen über die Umgebung wahrnehmen kann. Dazu zählt die Lage- und Bewegungserkennung.

Die Sensoren kommen in vielen Anwendungen zum Einsatz: Navigationssoftware, wie Google Maps, richtet die Karte nach Norden aus. Durch den Einsatz von Sensoren bei Computerspielen wird dem Benutzer eine wesentliche intuitivere Steuerung bereitgestellt. Auch der Wechsel zwischen Portrait- und Landscape-Modus, bei der Neigung des Smartphones, erfolgt über die Lagebestimmung durch Sensoren im Gerät.

Weitere Sensoren stellen verschiedene Umgebungsbedingungen wie Temperatur, Umgebungsdruck und Lichtverhältnisse zur Verfügung.

Die Sensoren liefern die Rohdaten mit einer hohen Genauigkeit, die über eine Schnittstelle in der Software mit einer geringen Abtastrate abgerufen werden können.

Die Qualität der Sensordaten ist stark von der Genauigkeit, Auflösung, der Abtastrate und dem Rauschen des physikalischen Sensors abhängig.

Nicht jeder Sensor ist tatsächlich physikalisch im Gerät verbaut, sogenannte virtuelle Sensoren, oder auch Software-Sensoren genannt, liefern durch die Software berechnete Daten. Hierbei werden Messwerte von „echten“ Sensoren, die als Hardwarekomponente im Gerät verbaut sind, abgeleitet. Ein Beispiel eines virtuellen Sensors ist der Sensor zur Bestimmung der linearen Beschleunigung. Hierbei wird die tatsächliche Beschleunigung anhand des physikalisch verbauten Beschleunigungssensors ermittelt. Diese Daten werden jedoch durch die Schwerkraft beeinflusst. Die Software filtert die Schwerkraft heraus und stellt sie als virtueller Sensor zur Verfügung.

Android unterstützt insgesamt 13 verschiedene Sensoren. Dabei ist dem Hersteller überlassen, welche Sensoren tatsächlich im Gerät verbaut sind. Das bedeutet, dass einer auf Android basierenden Anwendung nicht unbedingt jeder Sensor zur Verfügung stehen kann. Das *Samsung Nexus S* besitzt beispielsweise keinen Sensor zur Ermittlung des Umgebungsdrucks und im *HTC Nexus One* ist kein Gyroskop verbaut [HTC].

Der Beschleunigungssensor, der Magnetfeldsensor und das Gyroskop sind Sensoren, mit denen die räumliche Lage des Gerätes abgeleitet werden kann.

Im Rahmen dieser Arbeit werden diese Sensoren zur Bestimmung der Lage eines Smartphones in einem drei-dimensionalen Raum eingesetzt. Tabelle 3 zeigt die wichtigsten Bewegungs- und Lagesensoren eines Android-Systems.

Sensor	**Sensortyp**	**Beschreibung**	**Verfügbar seit API-Level**
Beschleunigungssensor	Hardware	Beschleunigung des Geräts in m/s² (inkl. Gravity)	3
Gyroskop	Hardware	Rotation in rad/s aller Achsen	9
Lineare Beschleunigung	Software	Beschleunigung des Geräts in m/s² (exkl. Gravity)	9
Magnetisches Feld (Kompass)	Hardware	Geomagnetische Feld in µT	3
Orientierung	Software	Rotationsgrad des Gerätes	3
Rotationsvektor	Software	Rotation (einheitenlos)	9
Gravity-Sensor	Software	Schwerkraft die auf das Gerät wirkt in m/s²	9

Tabelle 3: Die wichtigsten Sensoren zur Bestimmung der Bewegung und Lage des Geräts

Die Umgebungssensoren liefern Messwerte wie z.B. die Temperatur in Grad Celsius. Bei den Bewegungs- und Lagesensoren stehen die Messwerte in einem Bezug zu einem drei-dimensionalen Koordinatensystem. Je nach Sensortyp steht die Relation der Daten in einem lokalen Geräte- oder Weltkoordinatensystem. Laut [Google01] sind beide Koordinatensysteme wie in Abbildung 9 (S. 23) definiert.

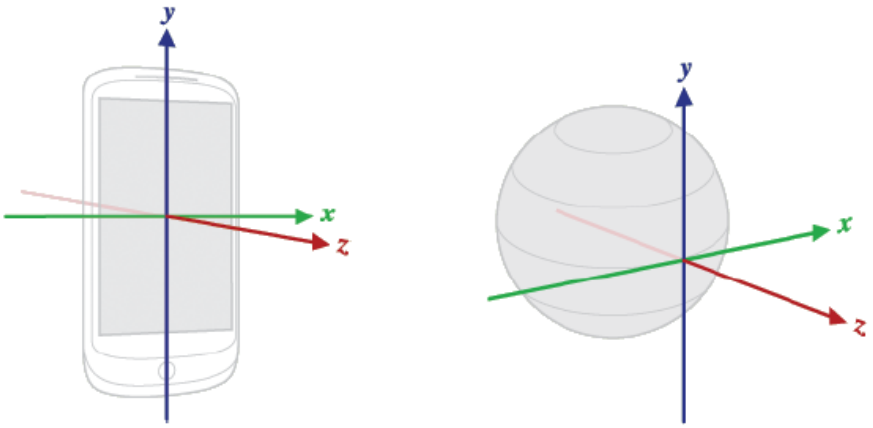


Abbildung 9: Device- und Weltkoordinatensystem

Das lokale Gerätekoordinatensystem, in der Literatur auch Device-Koordinatensystem genannt, ist relativ zum Display des Smartphones ausgerichtet. Die X-Achse verläuft horizontal zum Display und zeigt nach rechts. Die Y-Achse ist vertikal ausgerichtet und zeigt in Richtung des Lautsprechers des Smartphones nach oben. Die Z-Achse zeigt aus dem Display heraus.

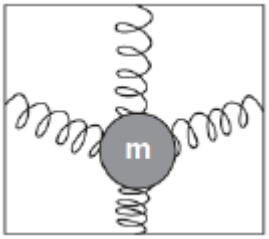
Das Weltkoordinatensystem, auch als globales Koordinatensystem bezeichnet, repräsentiert eine externe, sich nicht ändernde Referenz. Die Y-Achse zeigt in Richtung des magnetischen Nordens, während die X-Achse, daran ausgerichtet um 90 Grad zur Y-Achse, in Richtung Osten zeigt. Die Z-Achse zeigt vom Erdzentrum in Richtung Himmel.

Im folgenden Abschnitt wird ein Überblick über die für diese Arbeit relevanten Sensoren gegeben und in 2.12 werden diese Sensoren hinsichtlich ihrer Funktionalität zur Bestimmung der Lage eines Smartphones untersucht.

Beschleunigungssensor

Der Beschleunigungssensor (engl. Accelerometer) misst die aktuelle Beschleunigung für alle drei Achsen.

Die Messwerte werden von der Schwerkraft auf der Erde beeinflusst. Liegt das Gerät ohne sich zu bewegen auf einem Tisch, wird die konstante Erdbeschleunigung (Gravitation) von ca. 9,81 Metern pro Quadratsekunden gemessen.



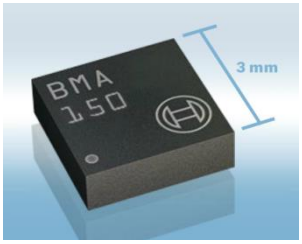


Abbildung 10: Beschleunigungssensor BMA150 von [Bosch]

Abbildung 11: Technisches Konzept eines Beschleunigungssensors, der eine konstante Erdbeschleunigung misst [Grabowski 2014]

Die Funktionsweise eines Beschleunigungssensors funktioniert nach dem Feder-Masse Prinzip (Abbildung 11). Hierbei wird die Beschleunigung anhand einer Testmasse registriert. Die Testmasse ist über eine elastische und verformbare Halterung im Sensor verbaut. Diese federähnliche Halterung wird durch eine Beschleunigung des Sensors verformt. Die Trägheitskräfte, die auf die Testmasse wirken, führen zu einer Verschiebung der Testmasse. Diese Verschiebung x ist proportional zur anliegenden Beschleunigung a : $x = \frac{M*a}{D}$ und wird auf allen Achsen im Bezugssystem des Sensors gemessen. Da die Federstärke D bekannt ist, kann die Beschleunigung anhand der Verschiebung der Testmasse M berechnet werden. Es existieren verschiedene physikalische Messprinzipien, um die Verschiebung der Testmasse zu ermitteln.

In modernen Smartphones werden häufig kapazitive Beschleunigungssensoren verbaut. Diese zeichnen sich durch eine fehlerarme Messung der Beschleunigung aus und können in großen Mengen und kostengünstig hergestellt werden. Bei den kapazitiven Beschleunigungssensoren wird der Abstand zweier Kondensatorelektroden (Kondensatorplatten) zur Bestimmung der Beschleunigung verwendet. Eine Elektrode wird als freie Testmasse realisiert. Die zweite Elektrode ist fest am Gehäuse des Sensors verbaut. Ändert sich der Abstand zwischen den Kondensatorelektroden

durch eine Beschleunigung des Konstrukts, verändert sich die Kapazität des Kondensators in einer linearen Abhängigkeit. Um die Genauigkeit der Messung zu erhöhen, werden beide Seiten der Testmasse mit einer Kondensatorelektrode ausgestattet. Man spricht dabei von Differenzialkondensatoren. Die Beschleunigung wird daraufhin anhand der Differenzialkapazität abgeleitet.

Der Beschleunigungssensor kann in Android in verschiedenen Frequenzstufen ausgelesen werden. Es ist eine Abtastrate von bis zu mindestens 50 Hz möglich. Je nach Gerät können höhere Frequenzen erreicht werden [Google1]. Tabelle 4 zeigt die in Android definierten Frequenzstufen zum Auslesen der Beschleunigungsdaten (Die Frequenzstufen wurden auf einem HTC Desire ermittelt).

Bezeichnung	Frequenz
NORMAL	ca. 4 Hz
UI	ca. 10 Hz
GAME	ca. 25 Hz
FASTETS	ca. 50 Hz

Tabelle 4: Frequenzstufen zur Abtastung der Sensordaten in Android

Magnetfeldsensor

Der Magnetfeldsensor, auch Kompass genannt, ermittelt die Ausrichtung des Smartphones zum magnetischen Feld der Erde. Der magnetische Nord- und Südpol stimmt nicht mit den geografischen Polen überein. Der Differenzwinkel wird als der Deklinationswinkel bezeichnet und muss insbesondere bei Anwendungen mit Bezug zum geografischen Norden, beispielsweise in der Navigation, berücksichtigt werden. Die magnetische Flussdichte wird in der SI-Einheit Tesla (T) gemessen. Die Funktionsweise des Sensors nutzt das Prinzip des Hall-Effektes aus. Der Hall-Effekt beschreibt das Auftreten einer Spannung, wenn ein Magnetfeld auf einen Stromleiter wirkt. Die Spannung steht dabei im selben Verhältnis zur Stärke des magnetischen Feldes.

$$U_H = A_H \frac{\mathrm{IB}}{\mathrm{d}} \qquad (1)$$

Die Spannung U_H lässt sich aus der Stromstärke *I*, der magnetischen Flussdichte *B*, der Dicke der Probe *d* und einer Materialkonstante A_H nach der oben aufgeführten Formel berechnen (1).

Um Aussagen über die Lage des Smartphones zu treffen, wird das Magnetfeld auf drei orthogonalen Achsen gemessen.

Die Messergebnisse des Magnetfeldsensors sind stark von äußerlichen Umwelteinflüssen abhängig und somit sehr empfindlich auf Störungseinflüsse.

Gyroskop

Das Gyroskop, auch als Kreiselinstrument bezeichnet, ermittlet die Winkelgeschwindigkeit der drei Achsen bei einer Drehung des Smartphones. Die Winkelgeschwindigkeit wird in Radiant pro Sekunde gemessen (rad/s).

Der Name Gyroskop (griech. Gyros, Kreisel) stammt von dem alten technischen Konzept der Kreiselinstrumente, bei dem nach dem physikalischen Erhaltungssatz einer rotierenden Masse (Kreisel), die Winkelgeschwindigkeit ermittelt werden kann. Diese Technik würde aus Platzgründen nicht in ein Smartphone passen. Daher werden heutzutage MEMS-Gyroskope verwendet, die auf der sogenannten Corioliskraft basieren.

Dabei werden mikromechanische lamellenartige Strukturen, die mit einem Plattenkondensator verbunden sind, in Schwingung versetzt. Aufgrund der entstehenden Corioliskraft verändert sich die Struktur und damit die Kapazität bei einer Drehung

um die Querachse. Aus der Kapazitätsveränderung des Plattenkondensators wird die Drehrate berechnet. Gyroskope dienen also dazu, die Drehbewegung des äußeren Bezugssystems zu messen. Dabei werden sie nicht von der Schwerkraft beeinflusst, was einen Vorteil gegenüber den Beschleunigungssensoren darstellt, da bei diesen erst die Schwerkraft rechnerisch herausgefiltert werden muss.

Das Gyroskop bestimmt die relative Änderung von der vorhergehenden Position zur aktuellen Position. Um einen absoluten Drehwinkel zu ermitteln, müssen alle relativen Änderungen über die Zeit integriert werden.

Das Gyroskop ist in der Herstellung teurer als der Beschleunigungs- oder der Magnetfeldsensor. Aus Gründen der Kostenminimierung ist das Gyroskop nicht in jedem Gerät verbaut.

2.11 Qualität der Sensordaten

Die Qualität der verbauten Sensoren kann von Gerät zu Gerät erheblich abweichen. Kleine Messfehler entstehen durch mechanische Unvollkommenheiten der Sensoren und führen zu einem Rauschen der Messwerte.

Das Ausgangssignal des Beschleunigungssensors ist mit zufälligen Fehlern behaftet, welche die Genauigkeit der Messung beeinträchtigen.

Auch das Gyroskop ist nicht fehlerfrei. Ein Abdriften des Gyroskops entsteht durch das Rauschen der Messwerte. Werden diese kleinen Abweichungen nicht korrigiert und zur Richtungs- und Lagebestimmung aufsummiert, kann dies über die Zeit zu einer wesentlichen Abweichung führen – dem Gyrodrift.

Ein Testversuch mit einem Samsung S3 ergab einen Drift von *0,3* Grad pro Sekunde. Das Gerät wurde dabei bewegungslos auf einem Tisch platziert und der absolute Drehwinkel über drei Achsen über die Zeit aufsummiert. Der Testversuch zeigt, wie fehleranfällig das Gyroskop zur Bestimmung der Lage im Raum ist. Abhilfe schafft eine softwareseitige Filterung der Gyroskop-Daten. Abbildung 14 (S. 31) zeigt die fehlerbedingte Drehung eines Smartphones vor und nach einer Herausfilterung des Drifts.

2.12 Bestimmung der Lage (Sensor Fusion)

Im Kontext der Head-mounted Displays ist die Aufgabe einer Head-Tracking Komponente die Bestimmung der Kopflage des Benutzers, also wie sich dieser im Raum orientiert. Dabei können die Daten der im vorherigen Kapitel vorgestellten Sensoren verwendet werden. Der folgende Abschnitt beschreibt die Bestimmung der Lage eines Smartphones.

Da ein Sensor allein die Lage eines Gerätes nicht korrekt bestimmen kann, können die Informationen von drei verschiedenen Sensoren verknüpft werden (Beschleunigungssensor, Magnetfeldsensor und Gyroskop). Dieses Prinzip wird auch als „Sensor Fusion“ bezeichnet, da die Daten mehrerer Sensoren verrechnet werden.

Der Beschleunigungssensor misst alle Beschleunigungen inklusive der Schwerkraft. Daher kann dieser dazu verwendet werden, anhand der Richtung der Schwerkraft, die Richtung nach unten zum Erdmittelpunkt zu ermitteln. Dieser Vektor kann mit dem Gravity-Sensor ermittelt werden.

Der Gravity-Sensor ist ein virtueller Sensor, der die Richtung der konstanten Erdbeschleunigung ermittelt (Gravitationsvektor), also der Vektor zum Erdmittelpunkt. Hierbei werden die Messdaten des Beschleunigungssensors verwendet. Alle Beschleunigungen, die die konstante Erdbeschleunigung überlagern, werden herausgefiltert. Hierbei kommt ein Tiefpassfilter zum Einsatz.

Anhand des Gravitationsvektors lässt sich die Lage des Gerätes auf nur zwei Achsen bestimmen. Liegt das Gerät auf einem Tisch, ohne sich zu bewegen, zeigt der Gravitationsvektor entlang der Z-Achse (siehe Abbildung 9, S. 23, links). Die Drehung um die Z-Achse ist dabei unbekannt. Der wie ein Kompass funktionierende Magnetfeldsensor ermittelt diese fehlende Information und vervollständigt die Orientierung des Gerätes im Raum. Anhand des Magnetfeldsensors wird die Y-Achse ausgerichtet. Die Y- und Z-Achsen sind also mit Hilfe der Kombination beider Sensoren (*g* und *m*) zu bestimmen. Die fehlende X-Achse steht orthogonal zu beiden Achsen und kann durch das Kreuzprodukt ermittelt werden (3).

$$\vec{z} \coloneqq \frac{\vec{g}}{\|g\|} \qquad (2)$$

$$\vec{x} \coloneqq \frac{\vec{m} \times \vec{g}}{\|m \times g\|} \qquad (3)$$

$$\vec{y} \coloneqq \vec{z} \times \vec{x} \qquad (4)$$

Die Android API stellt für die Lageermittlung die Methode *getRotationMatrix()* bereit, die genau nach diesem Prinzip implementiert ist. Der Rückgabewert der Methode ist eine Rotationsmatrix in Relation zum Weltkoordinatensystem, das ein globales Koordinatensystem darstellt. Als Eingabe werden die Daten des Beschleunigungs- und Magnetfeldsensors benötigt.

Die Messwerte des Magnetfeldsensors sind ungenau, träge und weisen ein Signalrauschen auf. Auch der Beschleunigungssensor ist mit zufälligen Messfehlern behaftet. Das Gyroskop, das die Winkelgeschwindigkeiten für alle drei Achsen liefert, ist viel genauer und hat eine wesentlich kürzere Reaktionszeit bei schnellen Bewegungen. Ein Nachteil stellt jedoch der Gyro-Drift dar. Um das Abdriften des Gyroskops und die rauschenden Daten des Magnetfeldsensors zu vermeiden, wird das Gyroskop für Änderungen in kurzen Zeitintervallen verwendet, während der Magnetfeld- und der Beschleunigungssensor bei langsamen Änderungen unterstützen. Dies kommt einer Tiefpassfilterung der Signale des Beschleunigungs- und Magnetfeldsensors und einer

Hochpassfilterung des Gyroskops gleich. Abbildung 12 zeigt ein Blockschaubild des Prinzips der Sensor Fusion zur Bestimmung der Geräteorientierung [Lawitzki 2012].

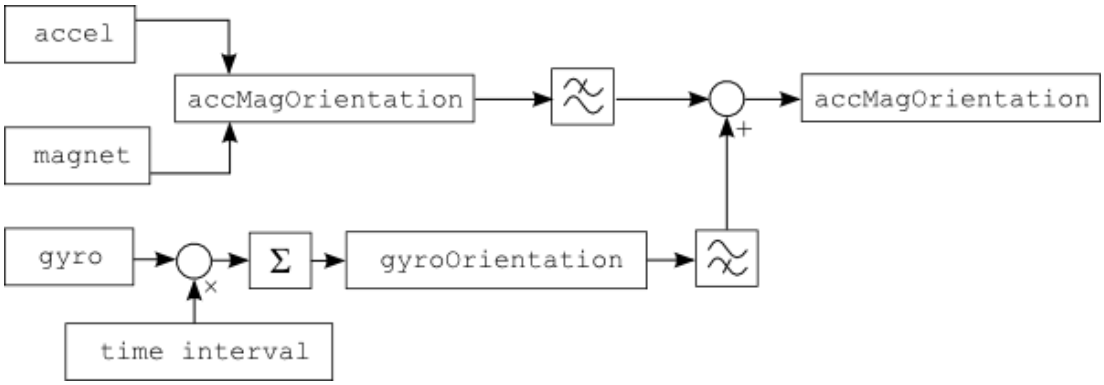


Abbildung 12: Sensor Fusion als Komplementärfilter

Die Sensoren liefern die Werte in festen Zeitintervallen. Die Werte können als ein Signal in einem Graph dargestellt werden, ähnlich wie bei einem Audiosignal. Der Tiefpassfilter glättet das verrauschte Signal des Beschleunigungs- und Magnetfeldsensors und eliminiert die winzigen und zufälligen Abweichungen in den Daten (Abbildung 13, S. 31). Dabei wird, jedes Mal wenn ein neuer Wert vom Sensor zur Verfügung steht, dieser mit einem Faktor (α) gewichtet. Der neue Wert (x_t) wird hierbei mit einem kleineren Anteil als der vorherige Wert (x_{t-1}) in das Endergebnis verrechnet. Der Gewichtungsfaktor wird in der Literatur auch als „Alpha-Wert" bezeichnet.

$$x_{TP} = (1 - \alpha) \cdot x_t + \alpha \cdot x_{t-1} \qquad (5)$$

Ein Tiefpassfilter lässt langsame Änderungen in einem Signal zu und eliminiert die schnellen Änderungen. Ein hoher Alpha-Wert (<1) führt zu einer langsamen Reaktionszeit, somit zu einem trägen Verhalten, dafür jedoch zu einer Glättung des Signals. Ein niedriger Alpha-Wert sorgt für ein umgekehrtes Verhalten. Bei einem Alpha-Wert von *0.2* fließen in das Ergebnis nur 20% vom aktuellen Sensor-Wert und 80% vom alten Wert ein.

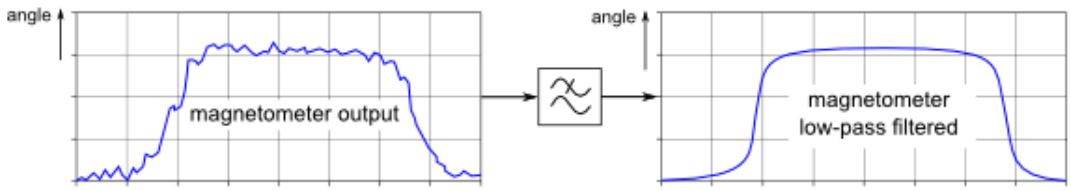


Abbildung 13: Tiefpassfilter

Das langsame Abdriften des Gyroskops durch das Aufsummieren von kleinen Fehlern in den Messwerten wird durch eine Hochpassfilterung kompensiert (Abbildung 14) und mit den gefilterten Werten des Beschleunigungs- und Magnetfeldsensors kombiniert.

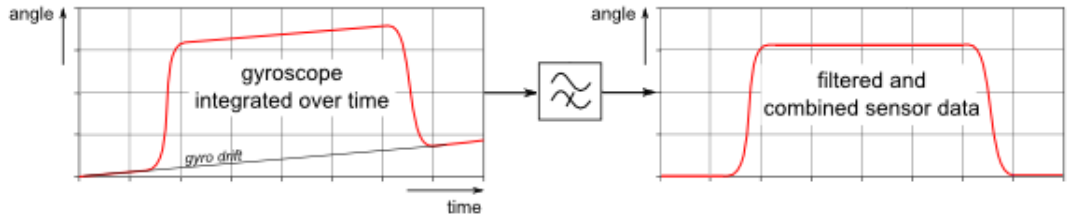


Abbildung 14: Hochpassfilter

Bei der Hochpassfilterung wird der herausgefilterte Hochfrequenzanteil mit den aktuellen Werten des Gyroskops (x_{Gyro}) ersetzt.

$$x_{Fused} = \underbrace{(1 - \alpha) \cdot x_{Gyro}}_{\text{Hochfrequenzanteil}} + \underbrace{\alpha \cdot x_{AccMag}}_{\text{Tieffrequenzanteil}} \qquad (6)$$

Dieses Filterprinzip wird auch als Komplementärfilter bezeichnet.

Die Implementierung eines Hochpassfilters und die Realisierung einer Sensor Fusion werden in 4.2.1 dieser Arbeit beschrieben.

Eine Alternative zum Komplementärfilter ist die Filterung der Sensordaten durch einen Kalman-Filter. Der Kalman-Filter ist ein Algorithmus zur Ermittlung der Sensorwerte und funktioniert, indem die Werte vorhergesagt werden und ein gewichteter Durchschnitt mit den am Sensor tatsächlich anliegenden Werten berechnet wird.

Der Kalman-Filter dient daher gut zum Entfernen von Störungen in einem Signal. Aufgrund der komplexen Implementierung und der ausreichenden Performance des

Komplementärfilters bzw. der von der Android API zur Verfügung gestellten Funktionalitäten, wurde dieser Ansatz in dieser Arbeit nicht weiter verfolgt.

2.13 Mathematische Repräsentation der Orientierung

Zur Winkeldarstellung der Lage im dreidimensionalen Raum werden in dieser Arbeit die Eulerwinkel verwendet.

Mit Hilfe der Eulerwinkel kann eine Orientierung eines Objektes im dreidimensionalen Raum beschrieben werden. Bei diesem Objekt kann es sich auch um die Kamera bzw. die Ausrichtung des Sichtfeldes handeln. Dabei wird die Lage des Objektes durch drei Winkel spezifiziert. Im Detail werden hierbei drei Rotationen um je einen Winkel auf einer Achse hintereinander durchgeführt und ergeben somit eine Gesamtrotation um alle drei Raumachsen. Die Reihenfolge der Einzelrotationen, in der sie ausgeführt werden, muss eindeutig festgelegt sein, da die Berechnung der Gesamtrotation nicht kommutativ ist und eine andere Reihenfolge somit zu einem anderem Rotationsergebnis führt. Grundlage dieser Arbeit ist die ZYX-Konvention für die Rotationsreihenfolge. Diese Konvention ist in der Luft- und Raumfahrt (DIN 9300) gebräuchlich.

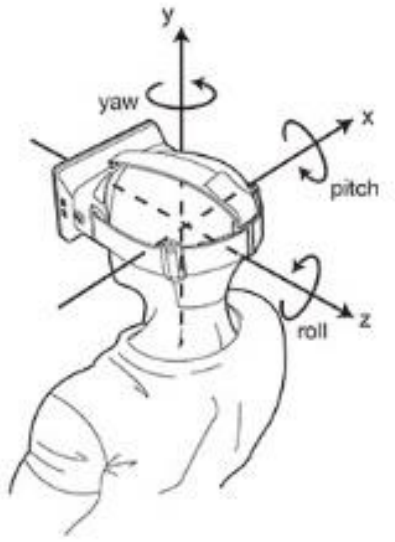


Abbildung 15: Yaw, Pitch und Roll beschreiben die Ausrichtung eines Kopfes (oculus.com)

Die drei Rotationswinkel werden als Gier- (Φ), Nick- (Θ) und Roll-Winkel (ψ) bezeichnet. Im Englischen werden diese *yaw*, *pitch* und *roll* genannt. Die Drehrichtungen der Winkel werden nach der „rechten Hand"-Regel definiert.

Die Rotation um einen Teilwinkel wird durch eine Drehmatrix berechnet (7) und kann nach (8) in eine Rotationsmatrix, welche die Gesamtrotation repräsentiert, überführt werden.

$$R_x(\Phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Phi) & -\sin(\Phi) \\ 0 & \sin(\Phi) & \cos(\Phi) \end{bmatrix} \qquad (7.1)$$

$$R_y(\Theta) = \begin{bmatrix} \cos(\Theta) & 0 & \sin(\Theta) \\ 0 & 1 & 0 \\ -\sin(\Theta) & 0 & \cos(\Theta) \end{bmatrix} \qquad (7.2)$$

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (7.3)$$

$$R_{RPY} = R_z(\psi)\, R_y(\Theta)\, R_x(\Phi) \qquad (8)$$

Ein Nachteil der Eulerwinkel ist das Auftreten des Gimbal-Lock in einem bestimmten Lagezustand, daher wird in der Computergrafik oft auf Rotationsberechnungen mit Quaternionen, die diesen Nachteil nicht aufweisen, gesetzt.

2.14 OpenCV Bibliothek

OpenCV ist eine frei zugängliche Software Bibliothek für die Bereiche Computer Vision und dem maschinellen Lernen (machine learning). Sie steht unter der BSD-Lizenz für die akademische und kommerzielle Nutzung frei zur Verfügung. OpenCV steht für Open Source Computer Vision und existiert seit 2006. Die Bibliothek verfügt über 2500 Implementierungen von verschiedenen Algorithmen.

OpenCV bietet den Entwicklern diverse Schnittstellen für C/C++, Python und Java an und unterstützt die Plattformen Windows, Linux, Mac OS, Android und iOS. Ein Wrapper[4] der OpenCV Bibliothek für Java steht auf GitHub[5] zur freien Verfügung. Diese Variante der Bibliothek kommt in dieser Arbeit zum Einsatz.

Der Java Wrapper unterstützt zusätzliche Funktionalitäten zur Aufnahme, Konvertierung und Streaming von Audio- und Videodaten (ffmpeg und ffserver) und der Ansteuerung der Xbox Kinect zur Gesichts- und Gestenerkennung. Kinect ist eine Steuerung der Videospielkonsole Xbox 360 von Microsoft.

OpenCV unterstützt Multi-Core Prozessoren und setzt den Fokus auf Echtzeit-Anwendungen und die Bildverarbeitung.

Die User-Community rund um OpenCV besteht aus mehr als 47 Tsd. Personen mit über 9 Mio. Downloads [OpenCV]. Das hauptsächliche Einsatzgebiet von OpenCV ist der Robotik-Bereich.

[4] Ein Stück Software, das ein anderes Stück Software umgibt.

[5] Hosting-Dienst für Software-Entwicklungsprojekte.

OpenCV besteht aus mehreren Modulen für die folgenden Anwendungsfelder:

- Gesichtserkennung
- Gestenerkennung
- 2D und 3D Merkmale (Region-of-Interest-Descriptoren)
- Mensch-Computer-Interaktion
- Objekterkennung
- Segmentierung und Erkennung
- Tiefenbilder (Stereoskopie)
- Ansteuerung von Bildverarbeitungsgeräten
- Fotogrammetrie
- Kalman-Filter zum Tracking von Objekten
- Maschinelles Lernen (Lernen eines Entscheidungsbaumes, Nächste-Nachbarn-Klassifikation, Bayes-Klassifikator, Neuronale Netze, Random Forest und Support Vector Machines)

2.15 Themenverwandte Arbeiten

Die Telepräsenz in einer entfernten realen Umgebung ist ein seit vielen Jahren untersuchtes Forschungsgebiet. Die fortschreitende Entwicklung der Netzwerk-Technologie hat die Idee der Telepräsenz vorangetrieben. Die Veröffentlichung von [Berestesky u. a. 2004] beschäftigt sich mit den dabei auftretenden Herausforderungen der Kommunikation im Bereich der Telerobotik.

Dieser Abschnitt soll einen Überblick über bestehende Ansätze und Arbeiten auf diesem Gebiet geben.

Im Rahmen einer Bachelorarbeit von Studenten der HTWG Konstanz (Martin Jehle und Frédéric Starnecker) wurde ein, wie in dieser Arbeit, ähnliches System realisiert ([Jehle 2014], [CT2014]). Dabei wurde die visuelle und auditive Wahrnehmung des Benutzers auf einen Drei-Achsen-Gimbal-Roboter ausgelagert. Die Fernsteuerung und Visualisierung erfolgt durch den Einsatz der *Oculus Rift*. Als Kamerasystem kommen zwei „Live!Cam"-Webcams mit einer 720p-Auflösung der Firma Creative zum Einsatz.

Abbildung 16: Bachelorarbeit an der HTWG-Konstanz (Quelle: Heise Verlag).

Martin Jehle arbeitet zurzeit (Stand: Feb. 2015) an einem zweiten Prototypen[6] mit nur einer einzelnen 2D-Kamera und erzeugt einen dreidimensionalen Eindruck durch das Prinzip des überlappenden Ausschneidens.

Ein weiterer interessanter Ansatz ist in der Arbeit von Nikolaus Lieb beschrieben [NLieb]. Die auf Kickstarter, eine Crowdfunding-Plattform zur Finanzierung von innovativen Projekten, präsentierte Arbeit, mit dem Titel „*Fly Like a Bird"*, verfolgt das Ziel, eine Kamera, die an einem Quadcopter montierte wurde, über ein HMD zu steuern und das Live-Bild aus der Luft auf dem Display zu visualisieren.

Ein interessanter Aspekt von [NLieb] ist der Ansatz zur Reduzierung der Reaktionszeit bei der Steuerung durch die Kopfbewegung des Anwenders. Die Reaktionszeit wird durch die Latenz der Datenübertragung beeinflusst. Die Kamera zeichnet ein

[6] http://movedreality.blogspot.de

etwas größeres Sichtfeld auf als das HMD darstellen kann. Dem Anwender wird ein kleinerer Ausschnitt präsentiert. Jede Kopfbewegung des Anwenders verschiebt den kleineren Ausschnitt des Sichtfelds im größeren Bild (Abbildung 17). Die Verschiebung, bei einer Kopfbewegung, wird von der Software des HMD übernommen. Nähert sich der Teilausschnitt dem Bildrand an, wird die Kamera bewegt. Dies soll eine virtuelle Kopfbewegung ergeben, die schneller auf die Bewegung des Anwenders reagiert.

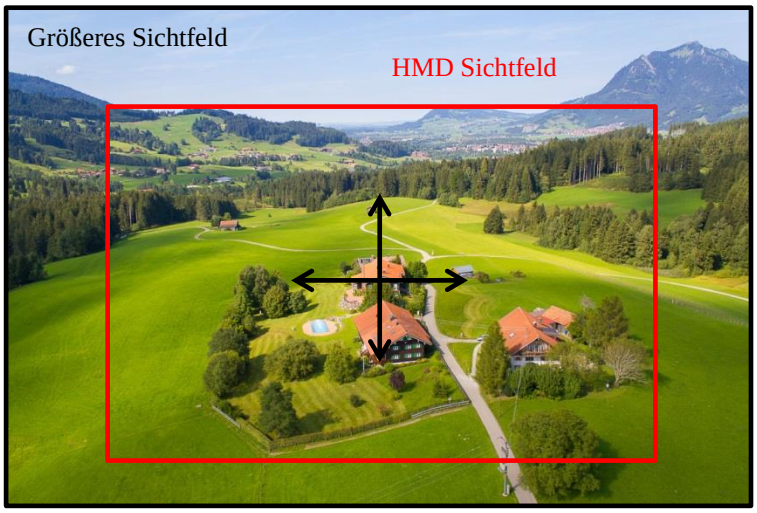


Abbildung 17: Ansatz zur Reduzierung der Bewegungsreaktionszeit durch das Verschieben eines Bildausschnittes

Bei den in diesem Abschnitt vorgestellten Arbeiten, welche die Bilder auf einem Head-mounted Display anzeigen, wird die Oculus Rift verwendet. Ein Ansatz, bei dem ein Smartphone als Head-mounted Display zum Einsatz kommt, wurde zum Zeitpunkt dieser Arbeit nicht gefunden.

Es existieren bereits zahlreiche kommerzielle Telepräsenzroboter auf dem Markt. Das Unternehmen iRobot entwickelt eine Robotik-Plattform, die mit verschiedenen Erweiterungen versehen werden kann. Die Plattform stellt eine Schnittstelle für Dritthersteller bereit und implementiert bereits die Funktionalität zur autonomen Navigation und dem Folgen von Personen.

Abbildung 18: Beispielhafte Aufbauten der iRobot AVA Mobile Robotics Platform [iRobot]

Der RP-7i Roboter der Firma InTouch Health aus den USA ist ein medizinischer Teleroboter, der es einem Arzt ermöglicht, mit seinen Patienten in Kontakt zu treten. Der bedienende Arzt steuert den Roboter aus der Ferne. Der Teleroboter lässt sich zudem mit medizinischen Geräten ausstatten, die aus der Ferne bedient werden können.

Abbildung 19: RP-7i Medizinroboter der Firma InTouch Health [RP7i]

3. Konzept

Die Hauptaufgabe dieser Arbeit ist die Entwicklung eines Telepräsenz-Systems. Die dafür benötigten Grundlagen sind bereits in dem vorherigen Kapitel beschrieben worden.

In diesem Kapitel soll der konzeptionelle Aufbau des im Rahmen dieser Arbeit entstandenen Systems erläutert werden. Die verwendete Hardware und die Entwicklung der Software werden im „Kapitel 4 – Implementierung“ beschrieben.

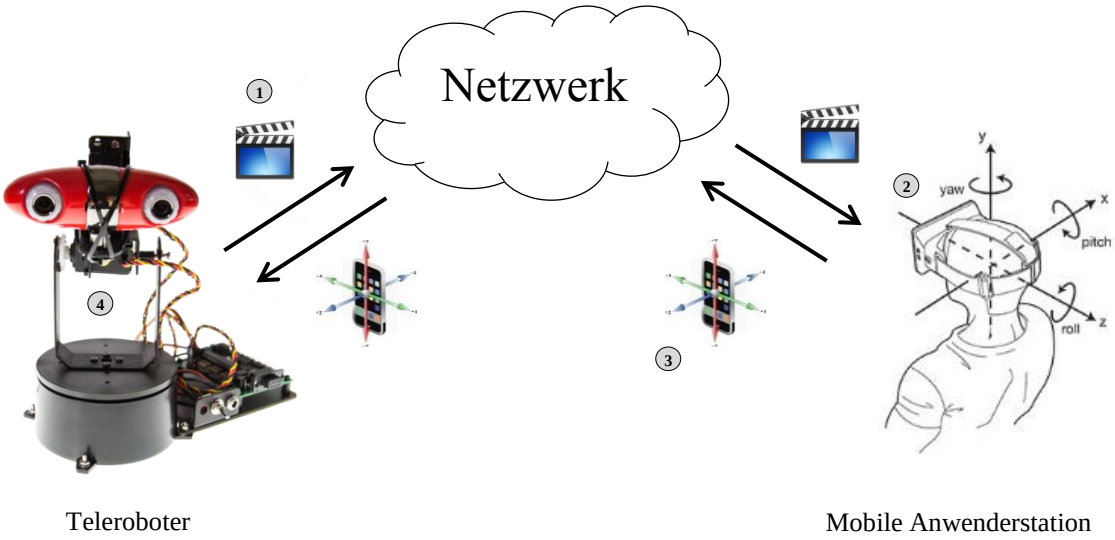


Abbildung 20: Verteiltes System bestehend aus zwei Komponenten und zwei wesentlichen Datenströmen

Das System besteht aus zwei Komponenten: Dem Teleroboter und der mobilen Anwenderstation. Die mobile Anwenderstation ermöglicht die Steuerung des entfernten Teleroboters durch den Benutzer. Hierbei wird die Kopfbewegung des Benutzers aufgezeichnet und durch den Teleroboter in der entfernten Umgebung wiedergegeben. Die visuelle Wahrnehmung des Teleroboters wird durch eine Kamera aufgenommen und von der mobilen Anwenderstation mittels eines Head-mounted Displays für den Benutzer visualisiert.

Es handelt sich hierbei um eine klassische Client- Serverarchitektur mit mehreren Datenströmen. Die Kommunikation des verteilten Systems erfolgt über ein lokales Netzwerk oder das Internet. Der stationäre Teleroboter kann somit an einem entfernten Ort stehen und aus der Ferne gesteuert werden. Abbildung 20 zeigt den Aufbau und die Kommunikation dieses Systems.

Der Teleroboter agiert als Serveranwendung und sendet die Bildinformationen der integrierten Kamera über das Netzwerk an die mobile Anwenderstation (1). Während das visuelle Abbild der entfernten Umgebung dem Benutzer auf dem Head-mounted Display präsentiert wird (2), werden die Kopfbewegungen (Head-Tracking) des Benutzers ermittelt und an den Teleroboter gesendet (3). Der Teleroboter empfängt diese Informationen und setzt sie auf ein um drei Achsen bewegliches Kamerasystem um (4).

Die mobile Anwenderstation, also das Smartphone, welches das HMD darstellt, kann als „Thin-Client“ verstanden werden, da dessen Aufgaben nur daraus bestehen, das Bild zu visualisieren und die Benutzereingaben (Head-Tracking) zu verarbeiten. Alle rechenintensiven Aufgaben werden vom Teleroboter übernommen, der eine höhere Rechenleistung zur Verfügung stellt. Dies macht den Einsatz eines Smartphones mit geringerer Rechenleistung erst möglich.

Die Kommunikation erfolgt in Echtzeit. Um einen möglichst hohen Immersionsgrad zu ermöglichen, bei dem der Benutzer denkt, er würde in die entfernte Umgebung eintauchen, muss der Videostream mit mindestens 30 Einzelbildern pro Sekunde und ohne eine Verzögerung (Live-Bild) die mobile Anwenderstation erreichen, wo das Bild dem Benutzer präsentiert wird. Das gleiche Kriterium gilt für die Übertragung der Kopfbewegung. Eine verzögerte Umsetzung der Kopfbewegung auf den Roboter führt dazu, dass sich die Kopfbewegung des Benutzers und die Bewegung der Kamera nicht synchron verhalten. Der Benutzer würde diese Verzögerung sofort wahrnehmen.

In den folgenden Abschnitten werden beide Komponenten – der Teleroboter und die mobile Anwenderstation - im Detail betrachtet.

3.1 Teleroboter

Die Komponente des Teleroboters ist ein steuerbares und stationäres Kamerasystem, das wiederum aus zwei Teilen besteht: Einem Kameraroboter und einer Steuereinheit zur Steuerung des Roboters.

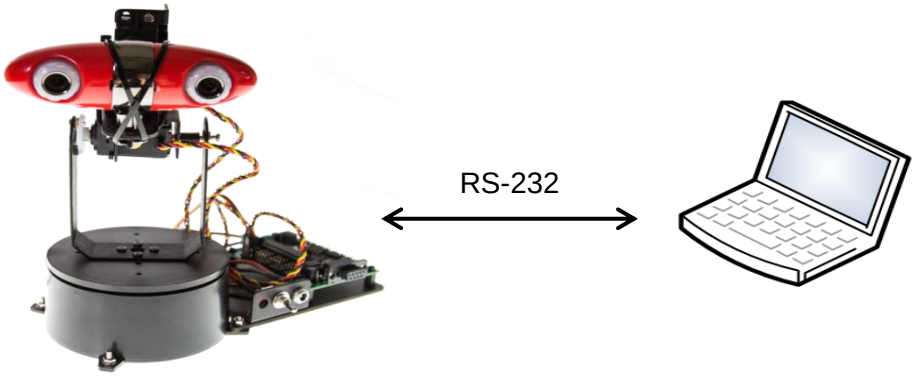


Abbildung 21: Aufbau des Teleroboters

Der Kameraroboter besteht aus einer stereoskopischen Kamera, drei Motoren zur Steuerung der Kamera in drei Richtungen und einem Servo-Controller, der die Motoren ansteuert. Die Kamera nimmt ein Bild pro Auge auf, also ein stereoskopisches Bild, das später für den 3D-Effekt sorgt (siehe 2.5).

Die Steuereinheit empfängt die Informationen zur Kopfbewegung von der mobilen Anwenderstation, verarbeitet diese und übermittelt die Steuerinstruktionen an den Servo-Controller, der die Motoren anhand der Head-Tracking Informationen ausrichtet. Die Bewegung der Kamera über drei Achsen soll der Kopfbewegung eines Menschen gleichen.

Die weiteren Aufgaben der Steuereinheit sind die Verarbeitung des bewegten Kamerabildes und der Bereitstellung des Videostreams für die mobile Anwenderstation. Bei der Steuereinheit handelt es sich um einen Rechner mit Software. Für diese Arbeit wird dazu ein herkömmlicher Laptop eingesetzt. Der Kameraroboter und die Steuereinheit kommunizieren über die serielle Schnittstelle (RS-232).

3.2 Mobile Anwenderstation

Die mobile Anwenderstation ist die Kontrollstation des Benutzers, sie dient zur Steuerung des Teleroboters und der Präsentation des Live-Videos, welches das in einer entfernten Umgebung stehende Kamerasystem aufzeichnet.

Dabei handelt es sich um ein Head-mounted Display, das durch den Einsatz von einem Smartphone realisiert wird. Der Kern des Head-mounted Displays ist eine mobile Anwendung für ein auf Android basierendes Smartphone.

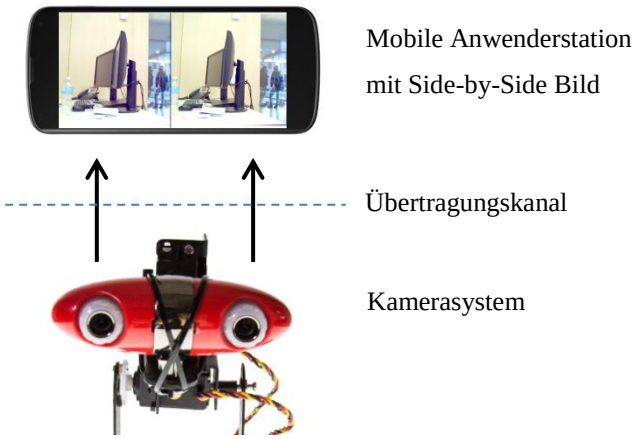


Abbildung 22: Side-by-Side Bild wird mittels stereoskopischer Kamera aufgezeichnet

Die Aufgabe der Anwendung besteht darin, das vom Teleroboter empfangene Live-Video als stereoskopisches Side-by-Side Bild zu visualisieren (Abbildung 22). Hierbei wird das Video, welches die rechte Kamera aufnimmt, auf der rechten Halbseite des Displays in der Anwendung dargestellt, dementsprechend das Video der linken Kamera in der linken Bildhälfte.

Abgesehen von der Anzeige der bewegten Bilder dient ein Head-mounted Display zur Ermittlung der Kopfbewegung des Benutzers, der das System auf seinem Kopf trägt. Die Kopfbewegung wird nach dem Prinzip der räumlichen Lagebestimmung eines Smartphones, wie es in den Grundlagen in Abschnitt 2.12 beschrieben wurde, bestimmt. Hierbei wird auf die in einem modernen Smartphone verbauten Sensoren zurückgegriffen. Diese Head-Tracking Informationen werden daraufhin in Echtzeit an den Teleroboter gesendet.

Für die Brillenhalterung kommt das Cardboard von Google zum Einsatz (2.8).

3.3 Head-Tracking und Gelenksystem des Kameraaufbaus

Die Position, Lage und Bewegungen des Kopfes werden durch drei Winkel dargestellt. Die drei Eulerwinkel…

- Yaw (Gieren)
- Pitch (Nicken)
- Roll (Rollen)

… beschreiben die Orientierung (2.13) des Head-mounted Displays und somit die Blickrichtung des Benutzers.

Das Gelenksystem, das die Kamera synchron zur Kopfbewegung ausrichtet, muss so gestaltet sein, dass es der Bewegung des menschlichen Kopfes nahe kommt. Dabei reicht das Umsetzen der menschlichen Halswirbelbewegung beim Umschauen mit drei Freiheitsgraden aus.

Abbildung 23 zeigt die Kopfbewegung auf drei Achsen anhand der menschlichen Halswirbelsäule.

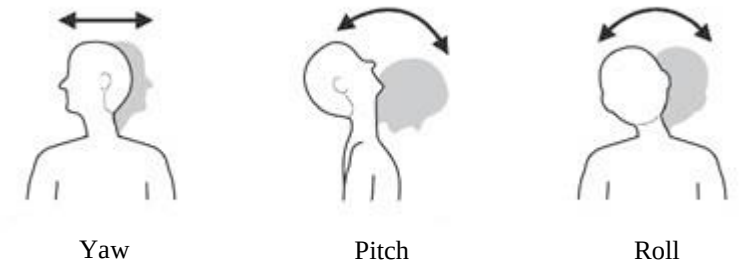


Abbildung 23: Head-Tracking

Der Yaw-Winkel repräsentiert die Kopfausrichtung beim Nach-links-und-rechts-Schauen, also der Drehbewegung des Kopfes. Pitch stellt die Bewegung des nach oben und unten Schauens dar. Der Roll-Winkel beschreibt die Kippbewegung des Kopfes auf die linke und rechte Seite.

Ein Vektor aus den drei Winkeln (Yaw, Pitch und Roll) wird an den Teleroboter gesendet und dort durch die Ausrichtung der ferngesteuerten Kamera als „virtuelle“ Kopfrichtung repräsentiert.

Das menschliche Genick besteht aus sieben Wirbelgelenken. Die Bewegung jedes einzelnen Gelenkes besitzt drei Freiheitsgrade [Schünkle 2000]. Insgesamt ergeben sich daraus 21 Freiheitsgrade. Die Nachbildung solch eines Gelenkes würde eine zu

große Herausforderung darstellen. So kann zur Nachbildung der Kopfbewegung nur ein auf drei Achsen bewegliches und somit drei Freiheitsgrade besitzendes Gelenksystem für die Ausrichtung der Kamera verwendet werden. Ein mit dem Kopf nach vorne Lehnen ist durch die Beschränkung auf drei Freiheitgrade nicht möglich.

3.4 Stereokamera

Eine Stereokamera bzw. stereoskopische Kamera besteht aus zwei einzelnen Kameras. Die beiden Kameras dienen zur Aufnahme der stereoskopischen Bilder für die räumliche Darstellung über das Head-mounted Display. Beide Kameras sind mit dem durchschnittlichen menschlichen Augenabstand voneinander ausgerichtet.

In dieser Arbeit kommt eine Kamera der Firma Minoru3D zum Einsatz. Da diese Kamera bereits zu Beginn dieser Arbeit zur Verfügung stand, wurde das Ziel verfolgt, diese Kamera in dem zu entwickelnden Prototyp einzusetzen.

Abbildung 24: Minoru3D Webcam

Im Laufe der Implementierung gilt es zu erläutern, ob diese Kamera den Anforderungen, die ein Telepräsenzsystem stellt, gerecht wird und ein Einsatz für den Teleroboter sinnvoll ist.

Die Minoru3D Kamera ist eine 3D-Webcam für den Desktopgebrauch. Sie liefert eine maximale Auflösung von 800×600 Pixeln pro Auge mit einer Frame Rate von 30 Bildern pro Sekunde.

4. Implementierung

In diesem Kapitel werden einige wichtige Aspekte der Implementierung herausgegriffen. Beschrieben wird die Entwicklung der wesentlichen Aufgaben des Teleroboters und der mobilen Anwenderstation. Ein essentieller Aspekt ist die Datenübertragung des Live-Videos und der Head-Tracking Daten.

Bei diesem Telepräsenzsystem handelt es sich um eine verteilte Anwendung. Ein Teil dieser Anwendung läuft auf der Steuereinheit, welche die Servomotoren und die Kamera des Teleroboters steuert. Der zweite Teil wird auf einem Smartphone ausgeführt.

Beide Softwarekomponenten kommunizieren über das UDP/IP Protokoll in einem WLAN-Netzwerk. Das drahtlose Netzwerk wird mit Hilfe eines WLAN-Repeaters aufgebaut.

Die Implementierung beider Komponenten wurde mit der Programmiersprache Java in der Version 1.7 umgesetzt. Hierbei wurde die Integrierte Entwicklungsumgebung Eclipse Luna 4.41 verwendet. Java ist eine plattformunabhängige Programmiersprache. Die Steuereinheit des Teleroboters kann somit auf verschiedenen Plattformen laufen. Bei dieser Arbeit kommt ein Windows 7 System zum Einsatz. Die mobile Anwenderstation (Android-Anwendung) basiert auf dem Android Developer Kit (Android 4.3.1).

Bei der Entwicklung des Systems wurde ein hoher Wert auf die Erweiterbarkeit der Software gelegt, da die Komponenten des Prototyps zu einem späteren Zeitpunkt, bei einer möglichen Weiterentwicklung des Projektes, austauschbar sein sollen. So kann leicht auf bessere Hardwarekomponenten und neue Funktionen gesetzt werden, ohne einen zu großen Eingriff im System vornehmen zu müssen. Während der Entwicklung konnten durch die Austauschfähigkeit der Komponenten einige Teile wie die Servomotoren und das Head-Tracking Modul durch simulierte Module (Mocks) als funktionierende Platzhalter innerhalb der Modultests verwendet werden. Dies erleichterte die Implementierung erheblich.

4.1 Teleroboter

Der Teleroboter besteht, wie im Konzept beschrieben, aus einer Hardwarekomponente (Kameraarm-Roboter) und einer Softwarekomponente, die auf einem Laptop (Steuereinheit) läuft.

4.1.1 Hardwareaufbau

Der Teleroboter besteht aus mehreren Hardwarebauteilen, die ein 3-Achsen-Gelenksystem zur Steuerung der Kamera zusammensetzen. Abbildung 25 zeigt den Teleroboter und hebt die einzelnen Bauteile durch die Nummerierung hervor.

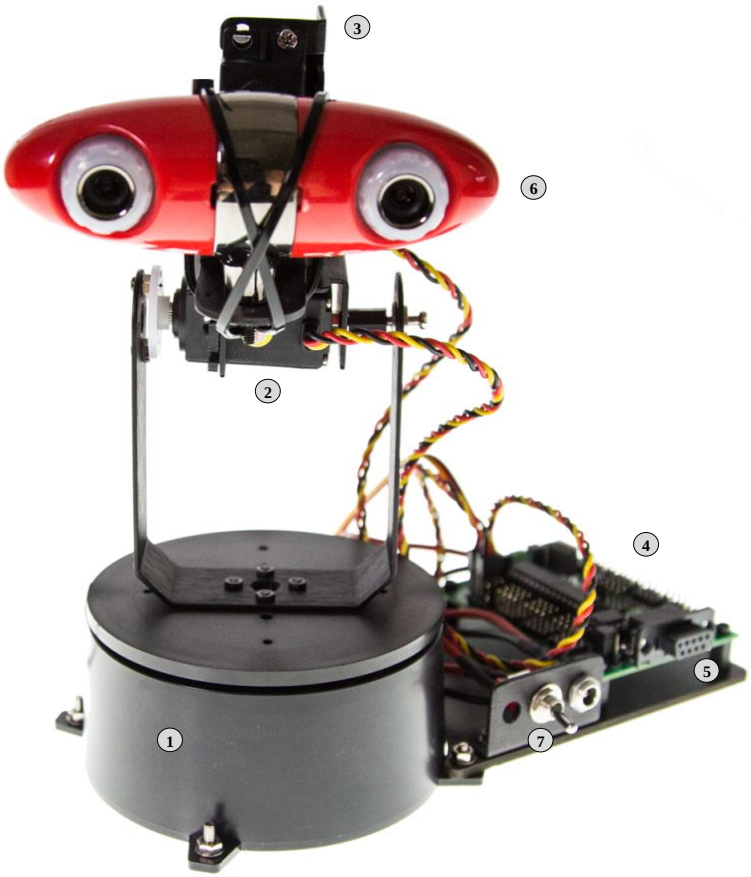


Abbildung 25: Hardwareaufbau des Teleroboters

Die Bewegung des Gelenkes erfolgt durch drei Servomotoren (1, 2, 3). Die Motoren werden über eine Steuereinheit, dem Servo Controller gesteuert (4). Dabei handelt es sich um einen Mikrocontroller, der die Steuerbefehle über die serielle Schnittstelle (5) entgegennimmt und die Servomotoren anhand der Steuerinstruktionen ausrichtet.

Eine detaillierte Beschreibung der Steuerung der Servomotoren wird in 4.1.2 erläutert.

Die stereoskopische Kamera ist auf dem Gelenkaufbau montiert und kann somit wie bei einer Kopfbewegung auf drei Achsen ausgerichtet werden (6).

Der Mikrocontroller und die Servomotoren werden mit einer Spannung von 6 Volt versorgt und können durch einen Schalter am Teleroboter geschaltet werden (7).

Folgende Aufzählung soll einen Überblick über die verwendeten Hardwarekomponenten des Teleroboters geben:

- Lynxmotion Robotik Bauteile für 3-Achsen Gimbal
- 3 Servomotoren der Firma Hi_{TEC}
- SSC-32 Servo Controller (RS-232)
- Minoru 3D Webcam (USB)
- Laptop als Steuereinheit zur Steuerung des Teleroboters

Der mechanische Aufbau des Roboters basiert auf dem AL5A Robotic Arm Combo Bausatz[7] der Firma Lynxmotion (Abbildung 26c, S. 48). Dieser Bausatz ist für einen Greifarm-Roboter konzipiert und wurde auf die Problemstellung dieser Arbeit, zur Realisierung des 3-Achsen-Gelenksystems, adaptiert. Für die Drehachse (Yaw-Winkel) kommt eine kugelgelagerte und servo-getriebene Tellerscheibe zum Einsatz (auf der vorherigen Seite in Abbildung 25 zu erkennen). Dieses kugelgelagerte Konstrukt hält die Kamerahalterung, welches sich durch einen Servomotor neigt (Pitch-Winkel) und durch einen dritten Servomotor nach links und rechts kippen lässt (Roll-Winkel). Um Vibrationen der Kamera bei der Gelenkbewegung zu vermeiden, ist sie mit zwei Kabelbinder an der Halterung fixiert.

Die visuelle Wahrnehmung wird durch die Minoru 3D Webcam realisiert (siehe 3.4). Die Kamera ist über die USB 2.0 Schnittstelle mit der Steuereinheit (Laptop) verbunden.

Um die Kamera simultan zur Kopfbewegung des Benutzers bewegen zu können, werden drei Servomotoren benötigt. Im Zuge dieser Arbeit fiel die Wahl auf die Servomotoren der Firma Hi_{TEC} vom Typ HS-645MG (Abbildung 26b).

[7] http://www.lynxmotion.com/c-124-al5a.aspx

Die drei Servomotoren mit doppelt kugelgelagertem Metall-Getriebe sind aus dem Modellbau bekannt und auch für den Bereich der Robotik einsetzbar. Jeder Servomotor leistet ein Drehmoment von 7,7 kg.cm, bei einer Spannung von 4,8 Volt. Der Servomotor ermöglicht eine Drehgeschwindigkeit von 60 Grad pro Viertelsekunde.

(a) (b) (c)

Abbildung 26: Mikrokontroller zur Steuerung der Servomotoren (a), HiTec Servomotor (b) und Bausatz von Lynxmotion (c)

Bei den Servomotoren handelt es sich um Analogservos. Die Ansteuerung der Motoren erfolgt über eine Pulsweitenmodulation. Die Winkelstellung wird dabei durch die Pulsdauer bestimmt. Abhängig von der Pulsdauer des PWM-Signals wird der Servomotor nach links bzw. rechts ausgelenkt. Eine Pulsdauer von 1,5 ms entspricht dabei einer Ausrichtung auf die Mittelstellung.

Die Ansteuerung der Servomotoren wird mit Hilfe des Mikrokontrollers (Servo Controller) realisiert. Mit Steuerbefehlen kann der Controller über die serielle Schnittstelle angesprochen werden und daraufhin das entsprechende Signal für die angeschlossenen Servomotoren generieren. Der Controller ist in der Lage, 32 Servomotoren über die Pins zu steuern.

Da einem modernen Laptop keine RS-232 Schnittstelle zur Verfügung steht, wird ein „USB-zu-RS-232“-Konverter eingesetzt.

Eine Kostenaufstellung zu den hier beschriebenen Hardwarekomponenten ist im Anhang zu finden.

4.1.2 Servosteuerung

Dieses Kapitel soll die Anforderungen an die Servomotoren erläutern, einen Überblick über die Ansteuerung der in dieser Arbeit verwendeten Hi_{TEC} HS-645MS Servomotoren geben und den Einsatz dieser evaluieren, indem ein Vergleich zu einem digitalen Servomotor der Firma Robotis gezogen wird.

Auf den Servomotor, der den Kameraaufbau nach oben und unten kippen lässt (Pitch-Winkel), wirkt die größte Last. Dieser Motor muss das Gewicht der Kamera, der Halterung und des Servomotors zur Neigung der Kamera (Roll-Winkel) tragen.

Die Servomotoren müssen eine flüssige und stufenlose Bewegung, trotz der auf den Motor wirkenden Last, ermöglichen.

Eine weitere Herausforderung bei der Wahl der richtigen Motoren stellt die Latenzzeit dar, denn nur, wenn eine schnelle Reaktionszeit und hohe Drehgeschwindigkeit der Servomotoren gewährleistet sind, um die Kamera schnell genug in die Zielposition auszurichten, kann dem Benutzer ein hoher Immersionsgrad bei Verwendung des Teleroboters ermöglicht werden.

Um die Komplexität der Implementierung der Servoansteuerung so gering wie möglich zu halten, sollte die Ansteuerung der Servomotoren eine einfache Steuerungslogik bzw. eine Schnittstelle in Java zur Verfügung stellen.

Die Hi_{TEC} HS-645MS Servomotoren werden durch den Servo Controller über die serielle Schnittstelle mit Befehlen im ASCII Format angesteuert. Dabei kann ein Servomotor per Befehl in eine bestimmte Stellung bewegt oder die aktuelle Stellung des Servomotors ausgelesen werden.

Die Ausführung eines Befehls wird durch ein Return-Zeichen (ASCII Code 13) angestoßen.

Sollen mehrere Servomotoren eine parallele Bewegung ausführen, können die Bewegungsbefehle gruppiert werden. Das Return-Zeichen markiert das Ende der Gruppierung und löst die simultane Bewegung der Servomotoren aus. Dezimalzahlen in den Befehlen, wie die Zielposition, werden als ASCII Strings der numerischen Ziffern dargestellt.

Die Befehle sind nach folgender Syntax aufgebaut [SSC32]:

 # <ch> P <pw> S <spd> <cr>

Der Beginn eines Befehls wird durch das Raute-Zeichen (#) mit einer darauf folgenden Kanalnummer <*ch*> markiert. Die Kanalnummer, die den Servomotor identifiziert, repräsentiert einen der 32 Pins, an denen die Servomotoren angeschlossen sind.

Beginnend mit einem ASCII „P“ kann die Pulsdauer in Mikrosekunden <*pw*> und somit die Ausrichtung des Servomotors nach dem Prinzip in Abbildung 27 beschrieben werden. Ein Wert von 500 bis 2500 ist dabei möglich.

Die Geschwindigkeit der Servobewegung kann mit einem „S“, gefolgt von der Anzahl der Umdrehungen pro Sekunde <*spd*>, bestimmt werden.

Die Dauer einer ganzen Bewegung kann durch *T* <*time*> bestimmt werden. <*time*> entspricht der Zeit in Millisekunden.

Alle Parameter, außer die Kanalnummer und die Pulsdauer, sind optional und können in einer beliebigen Reihenfolge im Befehl vorkommen.

Das Return-Zeichen <*cr*> (carriage return) schließt den Befehl ab.

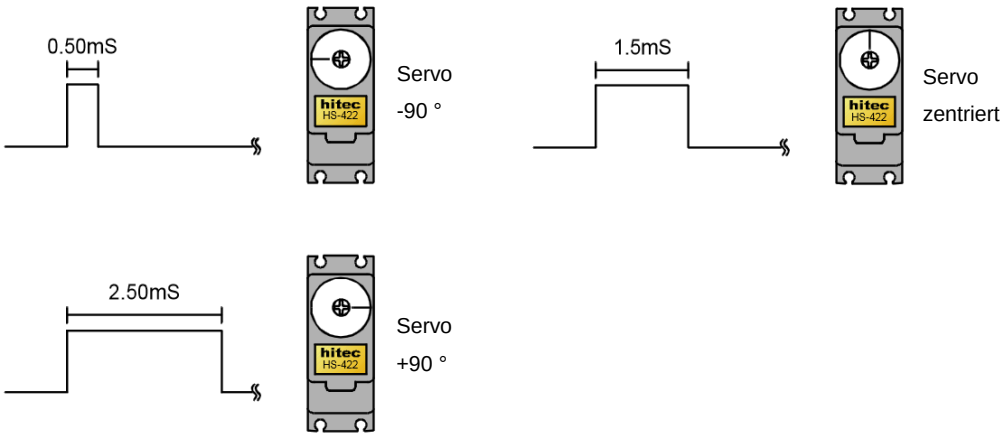


Abbildung 27: Pulsweitenmodulation der Hitec Servomotoren [SSC32]

Folgendes Beispiel zeigt den Befehl zur Bewegung des an Pin 2 angeschlossenen Servomotors nach Position 1500, was der Mittelstellung (0°) entspricht. Die ganze Bewegung dauert eine Sekunde:

 # 2 P1500 T1000

Die Gruppierung von mehreren Befehlen zur simultanen Ausführung erfolgt nach folgender Syntax:

<ch s1> P <pw> … # <ch s2> P <pw> … <cr>

Mehrere mit einem Raute-Zeichen (#) eingeleitete Befehle werden mit dem Eintreffen des Return-Zeichens <cr> vom SSC-32 Board zeitgleich ausgeführt.

Nach diesem Prinzip erfolgt die Ansteuerung des Gelenks der Kamerahalterung, die dieser Arbeit zugrunde liegt. Dabei werden die drei Servomotoren mit den drei Eulerwinkeln (Yaw, Pitch und Roll) simultan ausgerichtet.

Die Benutzung der seriellen Schnittstelle erfolgt über die RXTX Java Communication API.

Die Hi_{TEC} HS-645MS Servomotoren besitzen eine mechanische Sperre und lassen sich somit nicht um volle 360° drehen. Ein Servomotor dieses Typs ermöglicht bei einer Pulsdauer von 500 bis 2500 Mikrosekunden einen Drehwinkel von -90° bis +90°.

Dies bedeutet, dass die Kamera des Teleroboters auf eine Drehung von 180° auf allen drei Achsen beschränkt ist.

Eine Aufhebung der Drehachsensperre des Servomotors ist durch einen Eingriff in das Innenleben des Motorgetriebes möglich, dies ist jedoch nicht bei dieser Arbeit realisiert worden.

Eine Alternative zu den Hi_{TEC} HS-645MS Servomotoren stellen die digitalen Robotis Dynamixel Servomotoren dar. Das im Kapitel 2.15 erwähnte Projekt (Bachelorarbeit an der HTWG Konstanz) verwendet die Dynamixel Servomotoren von Robotis [Jehle 2014].

Digitale Servomotoren sind in der Regel besser verarbeitet, teurer und benötigen mehr Strom als Analogservos. Der Stromverbrauch der Digitalservos ist höher, da die Steuerelektronik bereits bei kleinen Positionsabweichungen voll gegensteuert. Die analogen Servomotoren weisen hierbei ein weicheres Verhalten auf.

Unter Last ist die Genauigkeit der Stellung bei digitalen Servomotoren meistens besser. Da das Kamerasystem dieser Arbeit ein Gewicht von wenigen Gramm hat, wird eine hohe Genauigkeit der Servomotoren bei großer Last nicht benötigt. Die analogen Servomotoren von Hi_{TEC} ermöglichen eine ausreichend feine Bewegung des Kameraaufbaus.

Die Robotis Dynamixel Motoren leisten eine Geschwindigkeit von 59 Umdrehungen pro Minute, bei einem Betriebswinkel von 0° bis 300° und einer Betriebsspannung von 12 Volt. Die bei dieser Arbeit verwendeten Hi_{TEC} HS-645MS Servomotoren er-

reichen eine Geschwindigkeit von 42 Umdrehungen pro Minute. Der Betriebswinkel hat durch die Getriebesperre einen Betriebswinkel von nur 180°.

In Bezug auf das Prototyping eines mechanischen Aufbaus bieten die Dynamixel Motoren den Vorteil, dass sich mehrere Servomotoren zusammenstecken lassen.

„Sie sind klein und lassen sich wie beim Legostein-Prinzip einfach miteinander kombinieren.“ [Jehle 2014, S. 16].

Die Dynamixel Servomotoren werden wie bei den Hi_{TEC} Servos über ein Controller Board angesprochen. Dabei erfolgt die Ansteuerung über den USB-Bus und ist erheblich komplexer als bei den Hi_{TEC} Servos. Ein Servo wird über eine eindeutige ID angesprochen, die zuvor fest in ein EEPROM geschrieben werden muss. Die Instruktionen zur Steuerung des Servos werden in eine „control table“ im Binärformat, samt Präambel und Checksumme, geschrieben [Jehle 2014]. Der Befehlssatz ist um einiges mächtiger und ermöglicht das Abfragen zusätzlicher Informationen wie der Last und Betriebstemperatur und dem Einrichten von Alarmen bei Überschreitung dieser Werte.

	Hi_{TEC} HS-645MS	Robotis Dynamixel
Preis	43 €	46 €
Drehmoment	7,7 kg.cm	1,52 kg.cm
Geschwindigkeit	42	59
Betriebswinkel	0° - 180°	0° - 300°
Auflösung	0,24°	0,29°
Betriebsspannung	6 V	12 V
Modulation	Analog	digital
Ansteuerung	Controller Board	Controller Board (USB)
Gewicht	55,2 g	54,6 g
Maße	40 mm x 20 mm x 38 mm	32 mm x 50 mm x 40 mm
Getriebe	Metall	Kunststoff

Tabelle 5: Vergleich der Servomotoren, des in dieser Arbeit verwendeten HS-645MS und der Alternative, dem Dynamixel von Robotis

Tabelle 5 zeigt einen Vergleich beider Servomotoren. Preislich unterscheiden sich beide Servomotoren kaum und bewegen sich in einer Preisspanne von 50 Euro. Die hauptsächlichen Unterschiede liegen in der Ansteuerung, dem Drehmoment und dem niedrigeren Betriebswinkel bei dem HS-645MS.

Die Entscheidung für die Hi_{TEC} HS-645MS Servomotoren fiel auf Grund der trivialen Ansteuerung über die serielle Schnittstelle mit simplen ASCII Befehlen und der ausreichenden Leistung, die eine flüssige Steuerung der Kamera ermöglichen. Die Bauteile aus dem Robotik Bausatz von Lynxmotion, welche die Halterung und das Gelenk bilden, passen zu den Servomotoren von Hi_{TEC}. Die Montage der Servomotoren ist dadurch relativ einfach.

Ein Nachteil der Servomotoren ist der Betriebswinkel von 180°, dieser stellt keine optimale Lösung dar, da eine menschliche Wirbelsäule eine Kopfbewegung von 200° erlaubt.

4.1.3 Abbildung der Orientierung auf die Servomotoren

Der Wertebereich der drei Eulerwinkel, welche die Kopfbewegung des Benutzers spezifizieren, unterscheidet sich vom Wertebereich der Pulsdauer zur Stellung der Servomotoren. Die Pulsdauer gibt die Ausrichtung der Servomotoren von 500µs bis 2500µs an, während die Eulerwinkel die Orientierung auf einer Achse in einem Bereich von -90° bis +90° abbildet. Beide Werte stehen in einem Verhältnis zueinander.

Zur Synchronisierung der Stellung der Servomotoren mit der Kopfbewegung des Benutzers wird der Wertebereich der Eulerwinkel ($x_{min}\ bis\ x_{max}$) in den Wertebereich der Servomotoren ($y_{min}\ bis\ y_{max}$) nach einer Verhältnisgleichung überführt (9, 10, 11, 12).

$$x_\Delta = x_{max} - x_{min} \tag{9}$$

$$y_\Delta = y_{max} - y_{min} \tag{10}$$

$$z = \frac{v - x_{min}}{x_\Delta} \tag{11}$$

$$v_{mapped} = y_{min} + (z * y_\Delta) \tag{12}$$

Gleichung (11) führt den Wertebereich der Eulerwinkel in einen Bereich von *0,0* bis *1,0*. Daraufhin kann anhand dieses Verhältnisses der korrespondierende Wertebereich der Servomotoren berechnet werden (12). Die Implementierung dieser Umrechnung wird in Codeausschnitt 1 erläutert.

```
private int map(float value, float leftMin, float leftMax, int
rightMin, int rightMax)
{
   // Figure out how 'wide' each range is
   float leftSpan = leftMax - leftMin;
   float rightSpan = rightMax - rightMin;

   // Convert the left range into a 0-1 range
   float valueScaled = ((float)(value - leftMin)) /
(float)leftSpan;

   // Convert the 0-1 range into a value in the right range.
   int ret = (int)(rightMin + (valueScaled * rightSpan));

   return ret;
}
```

Codeausschnitt 1: Mapping von Orientierung der Eulerwinkel zu Servoeinheiten

4.1.4 Kameraansteuerung

Das Abgreifen des Videobildes der stereoskopischen Kamera wird durch die Software Bibliothek OpenCV realisiert. OpenCV wurde im Kapitel „Stand der Technik“ im Abschnitt 2.14 beschrieben.

OpenCV bietet eine Softwareschnittstelle zur einfachen Ansteuerung eines physikalischen Kameragerätes. Bei der vorliegenden Arbeit wird die Minoru3D Webcam verwendet (3.4). Diese Kamera besteht aus zwei separaten USB-Geräten, die über ein einziges USB 2.0 Kabel mit der Steuereinheit des Teleroboters (Laptop) verbunden sind.

Die Software der Steuereinheit greift das Videobild beider USB-Kamerageräte ab, die vom Betriebssystem als zwei eigenständige Webcams erkannt werden.

```
// enter thread 1 (one thread for each eye):

// access camera 1
frameGrabberEye1 = new OpenCVFrameGrabber(1);

// set fixed resolution of 320 x 240
frameGrabberEye1.setImageWidth(320);
frameGrabberEye1.setImageHeight(240);
frameGrabberEye1.setFrameRate(30);  // set max. fps

frameGrabberEye1.start();     // start grabbing images from webcam

while(!aborted)
{
   // grab single image from camera
   grabbedImageEye1 = frameGrabberEye1.grab();

   // proccessing image
}

frameGrabberEye1.stop();
```

Codeausschnitt 2: Abfrage der Kamera über die OpenCV Bibliothek

Codeausschnitt 2 zeigt wie einfach das Auslesen der Kamera mit Hilfe der von OpenCV zur Verfügung gestellten *OpenCVFrameGrabber* Klasse ist. Dabei wird Bild für Bild, in einer Schleife, von der Kamera gelesen. Der Konstruktor der Klasse erwartet eine ID der Kamera. Die vorhandenen Kameras können zuvor über eine Methode, die eine Liste der IDs zurückgibt, ermittelt werden (nicht im Codeausschnitt aufgeführt). Mit der Methode *grab()* kann ein Einzelbild von der Kamera gelesen und eine Instanz der Klasse *IplImage* zurückgegeben werden. Dieses Bildobjekt repräsentiert einen Container von Bildinformationen.

Die Steuereinheit des Teleroboters ermittelt die Einzelbilder beider Kameras, eine Kamera pro Auge, nach dem in Codeausschnitt 2 beschriebenen Prinzip. Damit die Einzelbilder in einem niedrigen Intervall abgefragt werden können und somit ein flüssiger Video-Stream realisiert werden kann, wird bei der Abfrage beider Kameras die nebenläufige Programmierung eingesetzt. Dabei kann die Abfrage der einzelnen Kameras, da sie unabhängig voneinander sind, je von einem Thread verarbeitet werden. Die parallele Abfrage beider Kameras ermöglicht einen deutlichen Performancegewinn.

Sind beide Einzelbilder ermittelt, werden diese in ein stereoskopisches „side-by-side“-Bild zusammengeführt und zur Filterung und Komprimierung weiterverarbeitet, bevor das Bild an die mobile Anwenderstation gesendet wird.

Das fertige Bild, bestehend aus einem Einzelbild je Auge, wird wie in 4.3.4 beschrieben, von einem dritten Thread über das Netzwerk an die mobile Anwenderstation, also an das Smartphone, gesendet.

Die Minoru3D Webcam verspricht eine maximale Auflösung von *800×600* Pixeln pro Einzelkamera (Auge). Das Abgreifen der Bilder konnte bei der Implementierung jedoch nicht realisiert werden, da das simultane Abgreifen nicht möglich war. Es stellte sich heraus, dass der intern verbaute USB-Hub der Kamera nicht die Leistung zur parallelen Abfrage bringen kann. Eine Reduktion der Auflösung auf *320×240* Pixeln je Auge ermöglichte das parallele Abgreifen beider Bilder [AML01].

4.2 Mobile Anwenderstation

Die Anwendung zur Steuerung des Teleroboters ist eine Android Anwendung. Sie basiert auf der plattformunabhängigen Programmiersprache Java.

In den folgenden Unterkapiteln werden zwei wichtige Aufgaben der mobilen Anwendung beschrieben: die Ermittlung der Head-Tracking Daten, bei der die Lage des Smartphones bestimmt werden muss, und die Korrektur der Verzeichnung durch die Linsen des Head-mounted Displays (HMD).

4.2.1 Lagebestimmung

Die Lage des Smartphones, welche die Blickrichtung des HMD repräsentiert (Head-Tracking Informationen), wird nach dem Prinzip der Sensor Fusion bestimmt. Dabei werden Informationen von mehreren Sensoren verrechnet. Die Grundlagen wurden bereits in 2.12 erläutert.

Die Programmierschnittstelle von Android stellt einen abstrahierten Zugriff auf die Lagebestimmung durch Sensor Fusion in Form von virtuellen Sensoren bereit. Einer dieser Sensoren ist der Rotationsvektor-Sensor, der die physikalischen Sensoren fusioniert (Beschleunigungs- und Magnetfeldsensor).

Da in dieser Arbeit das Google Cardboard als HMD verwendet wird und dieses über einen Schalter verfügt, der ein magnetisches Feld erzeugt, das den Magnetfeldsensor beeinflusst, kann der Magnetfeldsensor in dieser Anwendung nicht in die Bestimmung der Lage mit einfließen.

Aufgrund dieser Einschränkung wird bei der Implementierung der „Game Rotation Vector“-Sensor verwendet. Bei diesem Sensor handelt es sich ebenfalls um einen virtuellen Sensor, der wie der Rotationsvektor-Sensor funktioniert, jedoch bei der Bestimmung des Yaw-Winkels (Drehbewegung um die Gierachse) nicht den Magnetfeldsensor berücksichtigt. Hierbei werden die Daten des Beschleunigungssensors und des Gyroskops verwendet.

Da bei dem virtuellen Game Rotation Vector Sensor nicht der Bezug zum magnetischen Norden als Referenz des Yaw-Winkels verwendet wird, kann es zu einem erheblichen Drift um diesen Winkel kommen. Die anderen Drehwinkel werden durch die Daten des Beschleunigungssensors unterstützt.

Die Android-Implementierung des virtuellen Sensors verfügt bereits über eine Filterung, allerdings zeigte ein Test auf einem Samsung S3 Gerät, dass der Yaw-Winkel

einen Drift von 0,3 Grad pro Sekunde aufweist. Der Gyrodrift scheint von Gerät zu Gerät zu variieren [Schomburg 2013]. Dies führt auf die Qualität der verbauten Sensoren zurück. Ein Test mit einem Nexus 5 zeigte keinen spürbaren Gyrodrift.

Bei der Implementierung dieses Projektes wird der Gyrodrift, der trotz bereits vorhandener Filterung des virtuellen Sensors vorkommt, mit Hilfe einer softwareseitigen Hochpassfilterung herausgefiltert.

Folgender Codeausschnitt zeigt die Implementierung der Lagerbestimmung mit dem Game Rotation Vector in Android.

```
public void onSensorChanged(SensorEvent event) {

   // only use sensor data from GAME_ROTATION_VECTOR
   if (event.sensor.getType() ==
         Sensor.TYPE_GAME_ROTATION_VECTOR) {
      float[] orientation = new float[3];
      float[] rotMat = new float[9];

      // get rotation matrix representing the device orientation
         from sensor values
      SensorManager.getRotationMatrixFromVector(rotMat,
                                          event.values);

      // get Euler-Angles from rotation matrix
      SensorManager.getOrientation(rotMat, orientation);

      // convert to degree for yaw, pitch and roll
      yaw = Math.toDegrees(orientation[0]);
      pitch = Math.toDegrees(orientation[1]);
      roll = Math.toDegrees(orientation[2]);

      // precess angles
      // ...
   }
}
```

Codeausschnitt 3: Lagebestimmung in Android

Die Sensordaten werden mit Hilfe der Methode `getRotationMatrixFromVector` in eine Matrix überführt, welche die Rotation des Gerätes beschreibt. Anhand dieser Rotationsmatrix können die drei Eulerwinkel mit der Methode `getOrientation` bestimmt werden.

Die Lageinformationen werden daraufhin an die Steuereinheit des Teleroboters gesendet. Die Übertragung der Daten wird in 4.3.1 beschrieben.

Einen Nachteil bringt der Game-Rotation-Vector-Sensor mit sich, denn er steht erst ab der Android-Version 4.3 (API-Level 18) zur Verfügung. Frühere Versionen müssen den Magnetfeldsensor mit einbeziehen oder die Sensor Fusion selbst implementieren.

4.2.2 Verzerrung

Ein Head-mounted Display – bei dieser Arbeit kommt dazu ein Smartphone zum Einsatz – verfügt über zwei Linsen, die für einen größeren Blickwinkel sorgen. Das größere Sichtfeld sorgt für eine realistischere Wahrnehmung der dargestellten Umgebung (Immersion), da das flache Display nicht als eine davor montierte Fläche registriert wird.

Durch die Krümmung der Linsen wölbt sich das Bild, für den Betrachter entsteht eine kissenförmige Verzeichnung (Pincushion Distortion). Abbildung 28a zeigt den Verzerrungseffekt des Bildes, der durch den Einsatz der Linsen entsteht.

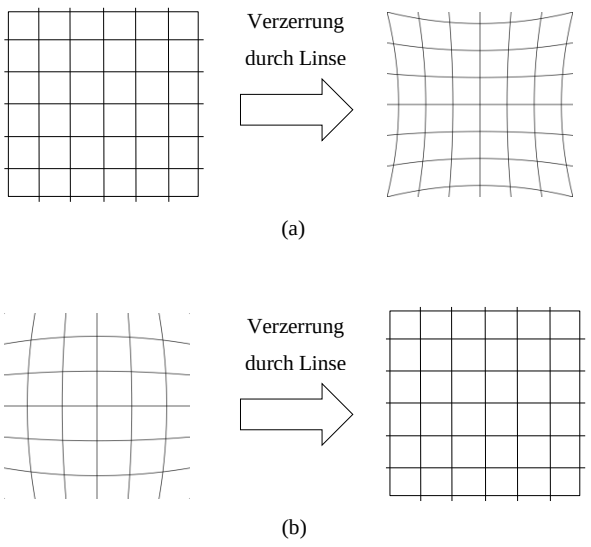


Abbildung 28: Korrektur der Linsenverzerrung durch Barrel Distortion

Um die durch die Linsen entstandene Krümmung auszugleichen, muss das Bild entgegen dieses Effektes verzerrt werden. Dabei kommt eine tonnenförmige Verzerrung des Originalbildes zum Einsatz (engl. Barrel Distortion). Abbildung 28b zeigt, wie der Linseneffekt das absichtlich verzerrte Bild zu einem unverzerrten Bild überführt.

Bei der tonnenförmigen Verzerrung nimmt die Stärke der Verzerrung der Bildelemente ab, je weiter das Bildelement vom Bildmittelpunkt entfernt ist.

In Abbildung 29 ist der Verzerrungseffekt zu erkennen, links unverzerrt und rechts verzerrt. Der Verzerrungseffekt ist am deutlichsten zu erkennen, wenn geradlinige Strukturen am Bildrand betrachtet werden.

Abbildung 29: Barrel Distortion (unverzerrt / verzerrt)

Die Bildverzerrung ist sehr rechenintensiv, da die Berechnung für jeden einzelnen Pixel erfolgt. Aus diesem Grund sollte der Algorithmus im Idealfall als Shader-Implementierung vom Grafikprozessor (GPU) ausgeführt werden. In dieser Arbeit wird die Berechnung der Verzerrung in Java von der CPU übernommen.

Folgende Formel zeigt das Berechnungsmodell der tonnenförmigen Verzerrung:

$$r_u = r_d(1 + kr_d^2)$$

r_u und r_d repräsentieren dabei den Abstand zum Mittelpunkt im unverzerrten und verzerrten Bild. Den Grad der Verzerrung bestimmt die Konstante k .

4.3 Netzwerkprotokoll und Datenströme

Wie bereits in Kapitel 3 beschrieben, bildet das Telepräsenzsystem ein verteiltes System, bestehend aus zwei miteinander kommunizierenden Softwarekomponenten.

Die Kommunikation lässt sich in drei verschiedene Datenströme aufteilen:

- Streaming des Live-Videos
- Streaming der Head Tracking Daten
- Steuerbefehle

Beim hier entwickelten System stellt die Kommunikation eine wesentliche Herausforderung dar. Ein Telepräsenzsystem erfordert hohe Anforderungen an das Echtzeitverhalten der Übertragung von Video- und Head Tracking Informationen. Nur eine performante Übertragung der Datenströme ermöglicht dem Anwender das Gefühl der Präsenz bei der Benutzung des Systems.

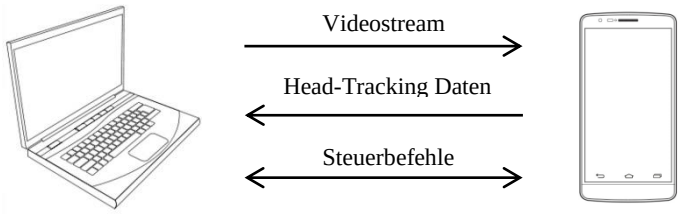


Abbildung 30: Datenströme der Anwendung und Richtung der Kommunikation

Abbildung 30 zeigt die Datenströme und deren Richtungen bei der Kommunikation. Die mobile Anwenderstation (Smartphone) sendet die Head Tracking Daten an die Steuereinheit des Teleroboters (Laptop), während die Steuereinheit den Videostream an die Anwenderstation übermittelt. Steuerbefehle zur Steuerung des Ablaufs der Anwendung werden in beide Richtungen verschickt.

Die Übertragung von Video- und Head Tracking Informationen werden über das UDP-Protokoll realisiert. Die Implementierung basiert auf der Java Socket Programmierung.

Das User Datagram Protocol (UDP) dient zur Kommunikation zwischen zwei Anwendungen und ist Teil der Transportschicht. UDP nutzt das Internet-Protokoll (IP), indem die UDP-Daten in das IP-Paket gepackt, nach den Regeln von IP übertragen und von der Gegenstelle wieder entpackt werden. Im Gegensatz zu dem äquivalenten TCP-Protokoll ist die Kommunikation über das UDP-Protokoll verbindungslos und

unzuverlässig. UDP-Nachrichten können verloren gehen. Es existiert kein Sicherungskonzept zur Sendewiederholung bei verlorengegangenen oder verfälschten Daten. Die korrekte Übertragung bzw. die Handhabung von fehlerhaften Nachrichten liegt in der Verantwortung des kommunizierenden Programms.

Da das UDP-Protokoll viel „schlanker“ ist (Sicherungskonzepte wie bei TCP sorgen für zusätzlichen Nachrichtenverkehr und Verarbeitungszeit) und die Übertragung den Echtzeitanforderungen gerecht werden soll, fiel die Wahl auf die Verwendung des UDP-Protokolls zur Übertragung beider Datenströme.

Ein Hauptgrund hierfür ist, dass immer nur die aktuellste Nachricht auf der Empfängerseite von Relevanz ist. Geht eine Nachricht bei der Übertragung verloren, ergibt es keinen Sinn, den Sendevorgang der verlorenen Nachricht zu wiederholen, da stattdessen immer die neusten Daten versendet werden können. Die Integrität der gesamten Übertragung stellt keine Anforderung an einen Live-Stream dar. Dies betrifft die Head Tracking Daten sowie die Einzelbilder des Live-Videos. Bei dem TCP-Protokoll müssen Nachrichten zwischengespeichert werden (buffering), um diese, falls ein Verlust festgestellt wurde, erneut zu senden. Dieser Mehraufwand verlangsamt die Übertragung.

Werden beispielsweise die Head Tracking Daten durch äußere Störungseinflüsse bei der Übertragung verfälscht oder gehen verloren, werden diese im nächsten Empfangszyklus, mit einem aktuelleren Zustand der Daten, überschrieben. Das Empfangen der Nachrichten in einer unterschiedlichen Reihenfolge (als diese versendet wurden), kann bei einer hohen Übertragungsrate, die bei Live-Streams vorausgesetzt ist, ebenfalls vernachlässigt werden.

Zur Übertragung der Datenströme wurde ein eigenes Nachrichtenformat implementiert, das auf die UDP-Nachrichten aufsetzt.

Eine Nachricht besteht aus zwei Datenfeldern: Einen Nachrichtentyp und den Nutzdaten (Payload).

Das Typenfeld hat eine Länge von vier Bytes (Integer) und beinhaltet einen numerischen Wert, der den Typ der Nachricht festlegt, um diese im Programm einer Funktionalität zuordnen zu können.

Die Nutzdaten können eine beliebige Länge einnehmen, jedoch nicht die maximale Länge eines UDP-Paketes überschreiten (inklusive dem Typenfeld), da die Nutzdaten nicht auf mehrere Pakete verteilt werden. Die maximale Länge eines UDP-

Paketes beträgt 65.507 Bytes [MS2014]. Eine Aufteilung der Nutzdaten auf mehrere UDP-Pakete wurde im Prototyp nicht implementiert (Siehe 4.3.2).

Folgende Nachrichtentypen sind implementiert:

- Typ 1: Head Tracking Daten
- Typ 2: Einzelbild des Live-Videos
- Typ 3: An- und Abmeldung
- Typ 4: Bestätigung der An- und Abmeldung (ACK)

Die Head Tracking Daten mit Nachrichtentyp 1 umfassen die Blickrichtung des Benutzers (4.3.1). Eine Nachricht des Typs 2 enthält ein von der Kamera aufgenommenes stereoskopisches Einzelbild (Frame), das wiederum aus einem Bild je Auge besteht (4.3.2).

Nachrichtentyp 3 und 4 sind für die Funktion der Mehrbenutzerfähigkeit reserviert. Eine Anwenderstation (Client) kann sich am Teleroboter anmelden und als Controller oder Zuschauer agieren. Die Kontrolle kann immer nur eine Anwenderstation zur gleichen Zeit übernehmen, eine Abmeldung übergibt die Kontrolle an eine zuschauende Anwenderstation.

Die Anmeldung und die Bestätigung der Anmeldung werden ebenfalls über das UDP-Protokoll übertragen. Da es sich hierbei um Steuerbefehle handelt, sollten diese in einer zukünftigen Weiterentwicklung über ein sicheres Protokoll, wie z.B. dem TCP-Protokoll, übermittelt werden. Da bei einer nicht erfolgreichen Anmeldung die Anmeldeprozedur wiederholt wird, kann dies im Prototyp dieser Arbeit vernachlässigt werden.

Jeder Nachrichtentyp wird von einer dazugehörigen Klasse verarbeitet. Die Klassen sind austauschbar. Die konkrete Verarbeitungsklasse der spezifischen Nachricht muss sich lediglich über die Empfangs-Controller Klasse (Logik des Nachrichtenempfangs) für einen Typ registrieren und wird daraufhin beim Empfang dieses Nachrichtentyps aufgerufen.

Dieses Prinzip macht die Kommunikation erweiterbar. Bei der Erweiterung der Anwendung können zukünftige Funktionalitäten somit leicht eingepflegt werden, ohne eine zu große Änderung an der Anwendung durchführen zu müssen.

Im Folgenden wird auf die Datenkanäle der Head Tracking- und Videodaten genauer eingegangen.

4.3.1 Head Tracking

Mit den Head Tracking Daten werden die Servomotoren des Kamerasystems synchron zur Position des Head-mounted Displays gehalten. Die Daten beschreiben dabei die Lage des Smartphones, die der Blickrichtung des Benutzers gleich kommt. Die Lage wird von den Eulerwinkeln repräsentiert, die von der Anwendung auf dem Smartphone ermittelt werden. In 4.2.1 wurde beschrieben, wie die Anwendung die Head Tracking Daten bestimmt.

Das Tupel, bestehend aus den drei Eulerwinkeln Yaw, Pitch und Roll, wird über das Netzwerk per UDP-Protokoll übertragen. Eine UDP-Nachricht der Head Tracking Daten besteht aus vier Attributen je 4 Bytes und hat eine Gesamtgröße von 16 Bytes (Abbildung 31).

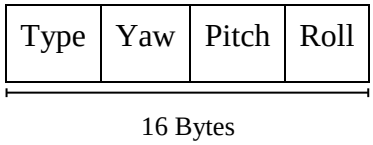


Abbildung 31: Head Tracking PDU

Die drei Eulerwinkel werden in Grad repräsentiert und nach dem Empfang der Steuereinheit des Teleroboters auf die Einheiten der Servomotoren umgesetzt (4.1.3).

Im Vergleich zur Übertragung des Live-Videos fallen die Head Tracking Informationen, aufgrund der geringen Größe der Nachricht, fast gar nicht ins Gewicht.

4.3.2 Videoübertragung

Die flüssige Übertragung des Live-Videos stellt gegenüber der Übertragung der Head Tracking Daten eine größere Herausforderung dar. Die Live-Übertragung erlaubt lediglich eine Latenzzeit von maximal 42 Millisekunden, um eine verzögerungsfreie Wiedergabe auf der Anwenderstation sicherzustellen, was einer Bildrate (FPS) von 24 Einzelbildern pro Sekunde entspricht.

Die Übertragung eines Einzelbildes, mit einer den Anforderungen entsprechenden Auflösung, stellt aufgrund der Größe der zu übertragenden Bildinformationen ein Problem dar. Die Lösung des Problems kann über eine Komprimierung der Videodaten realisiert werden.

Im Rahmen dieser Arbeit wurden zwei Ansätze zur Implementierung der Videoübertragung herausgearbeitet. Folgend werden beide Ansätze vorgestellt:

1. MPEG-Stream über das RTP-Protokoll (4.3.3)
2. Komprimierte JPEG‘s als Einzelbilder über das UDP-Protokoll (4.3.4)

Der erste Ansatz schien eine „optimale Lösung“ (4.3.3), konnte jedoch aufgrund von nicht ausreichender Unterstützung der MPEG-Codecs des Android-Systems und mangelhafter Dokumentation der verwendeten Frameworks nicht funktionsfähig implementiert werden. Eine tiefere Fehlersuche und Analyse der Ursache hätten den zeitlichen Rahmen dieser Arbeit gesprengt.

Die Videoübertragung des fertigen Prototyps, der in dieser Arbeit entstanden ist, wurde daher nach dem zweiten Ansatz realisiert.

4.3.3 MPEG-Stream über das RTP-Protokoll

Werden Video- und Audiodaten über das Internet an eine Anwendung eines hochskalierbaren Systems transportiert, kommt in vielen Fällen ein Streaming Server zum Einsatz. Ein Streaming Server sammelt Mediadaten verschiedener Quellen (z.B. Kameras) und überträgt diese zu verschiedene Clients, die als Abonnent des Streams agieren. Abbildung 32 illustriert den allgemeinen Aufbau eines verteilten Streaming Systems, bei dem ein Streaming Server zur Auslieferung von medialen Inhalten verwendet wird.

Bei dem hier realisierten Videostreaming-Ansatz kommt der „ffserver“ Streaming Server (b) von FFmpeg zum Einsatz. FFmpeg ist ein Multimedia-Framework zur Kodierung, Dekodierung und Streaming von Video- und Audiodaten.

Die OpenCV / JavaCV Bibliothek (2.14) bietet eine Java-Implementierung von Server (b) und Client (c). Der Streaming Server und ein Client der Videoquelle (a) ist eine Komponente der Steuereinheit des Teleroboters. Die Kodierung des Kamerabildes wird mit FFmpeg realisiert (a), während die Darstellung des Live-Videos über die JavaCV- Bibliothek auf dem Android-Gerät erfolgt.

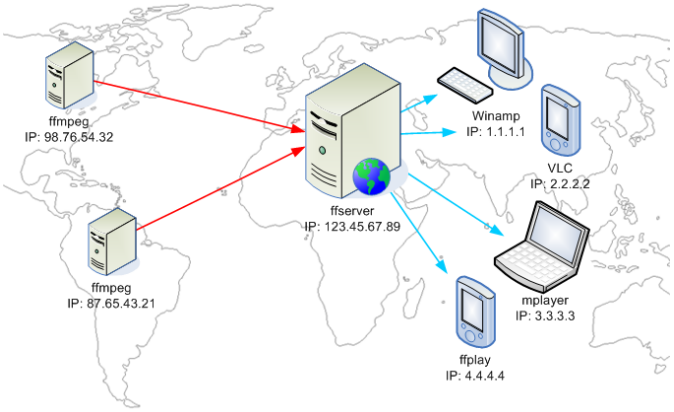


Abbildung 32: Streaming Client/Server Architektur (ffmpeg.org)

Die bewegten Bilder werden ohne Audiospur im H.264/MPEG-4 Standard übertragen. Der H.264 Standard gilt als ein hocheffizientes Kompressionsverfahren von Videodaten [TUB2012].

Die Übertragung der komprimierten Videodaten wird über das Real-Time Transport Protocol (RTP) realisiert. RTP ist ein IP-basiertes Netzwerkprotokoll, das in der Regel auf das UDP-Protokoll aufsetzt und der kontinuierlichen Übertragung (Streaming) von Audio- und Videodaten dient.

Der Einsatz eines Streaming Servers ermöglicht die einfache Skalierung der Teilnehmer. Ein neuer Client (mobile Anwenderstation) abonniert den Video-Feed und wird mit dem Live-Video Stream beliefert. Aufgrund dieses Vorteils sollte in dieser Arbeit die oben beschriebene Streaming-Architektur eingesetzt werden.

Dieser Ansatz konnte in dieser Arbeit nicht verwendet werden, da keine stabilen Voraussetzungen zur Videoübertragung ermöglicht werden konnten [SM01].

Auf einigen der getesteten Android-Geräten mit unterschiedlichen Versionen des Betriebssystems (Samsung S4 (4.4.2), S3 (4.3), Nexus 5 (4.3.1)) blieb die Anwendung beim Empfangen eines Einzelbildes (Frames) in einer Prozedur hängen. Die Anwendung wurde daraufhin vom Betriebssystem zwangsbeendet.

Ein alternativer Lösungsansatz ist der Empfang und die Darstellung des Video-Streams mittels HTML5-Technologien über eine in die Smartphone-Anwendung eingebettete Web-Browseransicht. Dabei kommt der JWPlayer[8] zum Einsatz. Dieser Lösungsansatz stellte sich ebenfalls als unbrauchbar heraus, da die Android-Implementierung bei der Vollbildunterstützung des Videos ein fehlerhaftes Verhalten aufweist.

„Taking video fullscreen causes devices to throw an error and stop.“ [JWPlayer] in Android 4.1+ (Jelly Bean).

Ein weiterer Lösungsansatz wäre die Videowiedergabe durch eine Drittanwendung auf dem Smartphone. So könnten der Empfang und die Darstellung über eine externe Videoplayer-Anwendung wie dem VLC-Player[9] erfolgen. Dieser Ansatz wurde in dieser Arbeit nicht umgesetzt. Lediglich die Wiedergabe des Streams wurde aus Testzwecken mit dem VLC-Player erfolgreich durchgeführt.

[8] http://www.jwplayer.com

[9] https://www.videolan.org/vlc/

4.3.4 JPEG-Frames über das UDP-Protokoll

In diesem Unterkapitel wird aufgrund des Scheiterns von Ansatz 1 (s. S.67) eine alternative Problemlösung der Videoübertragung beschrieben.

Jedes Einzelbild (Frame), das von der Kamera, wie in 4.1.4 beschrieben, ermittelt wird, wird in Form eines Byte-Arrays in eine UDP-Nachricht verpackt und über das Netzwerk an das Smartphone gesendet, das die mobile Anwenderstation bzw. das Head- mounted Display repräsentiert. Dabei wird das Einzelbild als JPEG-Bild kodiert und nach dem Empfang von der Anwendung zur Visualisierung des Bildes dekodiert.

JPEG ist ein Standard zur Kodierung von Bildinformationen. Die Bezeichnung führt auf das Gremium *Joint Photographic Experts Group* zurück, das den Standard im Jahr 1992 zum ersten Mal vorgestellt hat [JPEG]. Der JPEG-Standard beschreibt lediglich das Verfahren zur Bildkompression. Wie die komprimierten Bilddaten gespeichert werden, legt das JPEG File Interchange Format (JFIF) fest, das zum Austausch der Bilder dient.

Die Kodierung in das JPEG-Format ermöglicht eine Komprimierung der Bildinformationen. Abhängig von einem Kompressionsfaktor weisen die Bildinformationen eine unterschiedliche Größe auf. Die Komprimierung ist verlustbehaftet, das bedeutet, je größer der Kompressionsfaktor (Kompressionsstufe verhält sich entgegengesetzt zu diesem Faktor), desto weniger Informationen gehen bei der Komprimierung verloren und desto höher ist die Qualität des Bildes (Abbildung 33), was sich in der Größe der zu übertragenden Datenmenge bemerkbar macht. Codeausschnitt 4 (S. 70) zeigt die Komprimierung der Bildinformationen in das JPEG-Format.

Abbildung 33: JPEG-Bild einer Phalaenopsis mit von links nach rechts zunehmender Kompressionsstufe (abnehmender Kompressionsfaktor)

```
BufferedImage sideBySideImage = joinBufferedImage(
      grabbedImageEye1.getBufferedImage(),
      grabbedImageEye2.getBufferedImage());

Iterator iter = ImageIO.getImageWritersByFormatName("jpeg");
ImageWriter writer = (ImageWriter)iter.next();

ImageWriteParam iwp = writer.getDefaultWriteParam();
iwp.setCompressionMode(ImageWriteParam.MODE_EXPLICIT);
iwp.setCompressionQuality(0.5f);

ByteArrayOutputStream baos = new ByteArrayOutputStream();
writer.setOutput(new MemoryCacheImageOutputStream(baos));

IIOImage image = new IIOImage(sideBySideImage, null, null);
writer.write(null, image, iwp);
writer.dispose();
byte[] outputImage = baos.toByteArray();
```

Codeausschnitt 4: JPEG-Kodierung des Side-by-Side Bildes der beiden Roboteraugen

Das zu übertragende JPEG-Bild besteht aus den zwei Einzelbildern beider Augen des Teleroboters. Beide Bilder können somit synchron übertragen werden. Ein Bild je Auge hat eine Auflösung von *320 x 240* Pixeln. Demnach wird das Side-by-Side Bild mit der Gesamtauflöung von *640 x 480* Pixeln übertragen.

Das unkomprimierte Rohbild der Kamera wird in einem *BufferedImage*, einen Container für Bildinformationen, gehalten und mittels der Java-eigenen *`ImageWriter`* Klasse in das JPEG-Format kodiert. *ImageWriter* verfügt über diverse Parameter, wie den Kompressionsfaktor, der den Informationsgehalt und somit die Qualität des Bildes spezifiziert. Ein Java Output-Stream der Klasse *ByteArrayOutputStream* konvertiert das komprimierte Bild in ein Byte-Array, das über ein UDP-Socket an das Android-Gerät gesendet wird (nicht im Codeausschnitt dargestellt). Nach dem Empfang wird das Byte-Array auf dem Android-Gerät zurück in einen Bitmap-Bildcontainer konvertiert und auf dem Display dargestellt.

Das JPEG-Bild umfasst bei einem Kompressionsfaktor von *0,5* eine Größe von ungefähr *12.288* Bytes (abhängig von der Detailvielfalt in den Bildinformationen). Aufgrund der Maximalgröße einer UDP-Nachricht von 65.507 Bytes [MS2014] passt das JPEG-Bild in eine einzelne UDP-Nachricht. Wird ein größerer Kompressionsfaktor gewählt, überschreitet die Größe des JPEG-Bildes die maximale Größe einer UDP-Nachricht und muss auf mehrere Nachrichten verteilt werden. Eine Verteilung erhöht

den Datenverkehr und hat eine Reduktion der Bildrate (FPS), bei gleichbleibendem Datendurchsatz, zur Folge.

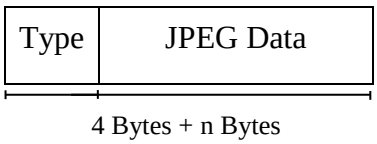


Abbildung 34: JPEG-Einzelbild PDU

Um das Protokoll zur Videoübertragung so einfach wie möglich zu halten, wurde bei vorliegender Arbeit auf das Verteilen der JPEG-Bilder auf mehrere Nachrichten verzichtet. Im lokalen WLAN kann eine über die Zeit konstante Bildrate von 30 Einzelbildern pro Sekunde erreicht werden. Theoretisch ist eine Bildrate von bis zu 40 Bildern pro Sekunde möglich. Da die Minoru3D Kamera allerdings auf 30 Bilder pro Sekunde limitiert ist, wird die Bildrate der Kamera bei der Übertragung der Einzelbilder nicht überschritten. Die Evaluierung hinsichtlich der Performance der Videoübertragung wird in 5.2 beschrieben.

Die Übertragung eines Videos in Form von mehreren JPEG-Einzelbildern wird oft in der Praxis unter dem MJPEG-Standard (Motion JPEG) bei der Videoübertragung von IP-basierten Webcams eingesetzt und ist dort ein verbreiteter Ansatz zur Kodierung von bewegten Bildern. MJPEG basiert häufig auf dem HTTP-Protokoll, bei der jedes Einzelbild separat kodiert und als HTTP-Stream übertragen wird. Die Einzelbilder werden in der HTTP-Nachricht durch einen String, der im MIME-Typ festgelegt wird, getrennt. Der empfangende Client (meistens ein Webbrowser) kann die Bilder somit korrekt als Video darstellen.

In dieser Arbeit wird kein Webbrowser zur Darstellung des Videos verwendet. Das Video wird Einzelbild für Einzelbild in der grafischen Benutzeroberfläche der Smartphone-Anwendung dargestellt (4.4.1). Abbildung 34 zeigt, wie ein Einzelbild separat in einer UDP-Nachricht versendet wird.

4.3.5 Vergleich der Ansätze

Die Vor- und Nachteile beider Ansätze werden in Tabelle 6 erläutert.

	Ansatz 1	Ansatz 2
Beschreibung	MPEG-Stream über das RTP-Protokoll	Stream aus JPEG-Bildern über das UDP-Protokoll
Videokodierung	H.264 / MPEG-4	JPEG pro Einzelbild
Übertragungsprotokoll	RTP / IP	UDP / IP
Implementierung	komplex	einfach
Dokumentation	nicht ausreichend, abhängig von Framework	sehr gut, da Java Standardklassen
Unterstützung von Android	Mangelhaft	volle Unterstützung
Kompression (30 Sek. Video mit 30 FPS)	6,2 MB	11 MB

Tabelle 6: Vergleich beider Ansätze zur Videoübertragung

Das JPEG-Kompressionsverfahren dient zur Komprimierung eines einzelnen Bildes und ist nicht für die Komprimierung von bewegten Bildern optimiert. Die einzelnen Bilder eines Videos werden unabhängig voneinander komprimiert. Ein Vorteil ist jedoch die einfache Implementierung. Die Programmierschnittstellen sind gut dokumentiert, da Java die Funktionalität zur JPEG-Kompression und der Socket Programmierung von Haus aus mitbringt.

Ein Verfahren zur Videokompression, wie es der H.264 Standard beschreibt, nutzt zur Kodierung zusätzlich die Ähnlichkeiten zwischen den Einzelbildern, die das Video ergeben. Das Kompressionsverfahren des ersten Ansatzes ist daher für die Kompression von Videodaten optimiert.

Ein Vergleich beider Ansätze (4.3.3 + 4.3.4) hinsichtlich der Kompression zeigt die Komprimierung eines 30-sekündigen Videos, welches mit einer Bildrate von 30 Einzelbildern pro Sekunde (FPS) aufgezeichnet wurde. Dabei hat ein Einzelbild eine Auflösung von *640 x 480* Pixeln mit einer Größe von *12.288* Bytes (JPEG-Kompressionsfaktor von 50%). Während das ganze Video im MPEG-4 Format eine

Datengröße von 6,2 MB aufweist, haben die 900 JPEG-Frames nach (13) eine Gesamtgröße von 11 MB (30 $frames * 30\ Sek. * 12.288\ Bytes$).

Formel 13 zeigt die Bestimmung der Datengröße (*size*) eines MJPEG-Videos in Bytes.

$$size = FPS * t * n \quad (13)$$

$FPS \coloneqq Anzahl\ Einzelbilder\ pro\ Sekunde$

$t \coloneqq Länge\ des\ Videos\ in\ Sekunden$

$n \coloneqq durchschnittliche\ Datengröße\ eines\ Einzelbildes\ in\ Bytes$

Aufgrund der einfachen Implementierung von Ansatz 2 und der Tatsache der problematischen Realisierung des ersten Ansatzes, fiel die Wahl auf die Übertragung des Video-Streams nach dem in 4.3.4 beschriebenen MJPEG-Prinzip.

4.4 Benutzerschnittstellen

Beide Komponenten dieser Arbeit (mobile Anwenderstation und die Steuereinheit des Teleroboters) verfügen über eine Benutzerschnittstelle zur Interaktion mit dem Benutzer und der Visualisierung.

Dieses Kapitel beschreibt die grafischen Benutzerschnittstellen des verteilten Systems.

4.4.1 Mobile Anwenderstation

Die mobile Anwenderstation zur Steuerung des Kameraroboters besteht aus einer Android-Anwendung, die über drei verschiedene Ansichten, im folgenden Views genannt, verfügt. Abbildung 35 (S.75) zeigt die Views der Anwendung und deren Lebenszyklen bei der Navigation.

Die Anwendung verfügt über einen Startup-Screen (Abbildung 35a), der zur Kalibrierung der Kamerabewegung des Teleroboters dient. Wird die Anwendung gestartet, beginnt der Kameraarm in der Ursprungsausrichtung (0° auf Yaw-, Pitch- und Rollwinkel). Das Beenden des Startup-Screens, durch den Toggle-Button des Google Cardboards (2.8), setzt die aktuelle Ausrichtung des Smartphones in Korrelation zu der Ursprungsausrichtung des Kameraarms. Somit kann die Ausgangsblickrichtung am realen Ort (Standort des Benutzers) durch den Benutzer bestimmt werden. Jede Bewegungsänderung des Smartphones führt zu einer sich analog verhaltenden Bewegung der Kamera.

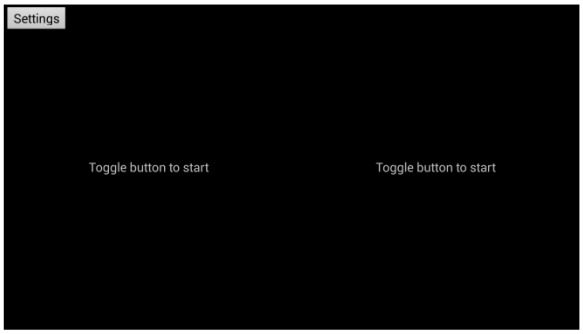
Die bewegten Bilder der Kamera werden als stereoskopisches Bild unter Anwendung der tonnenförmigen Verzerrung (4.2.2) visualisiert (Abbildung 35c). Wie in 4.3.4 beschrieben, hat das stereoskopische Bild eine Größe von *640 x 480* Pixeln. Auf einem Nexus 5 mit einer Full-HD Displayauflösung von *1920 x 1080* Pixeln wird das Bild hochskaliert, um die volle Auflösung auszunutzen. Darunter leidet die Bildqualität.

Über den Startup-Screen kann der Benutzer eine Konfigurationsansicht erreichen (Abbildung 35b). Der Benutzer ist in der Lage, folgende Parameter zu spezifizieren:

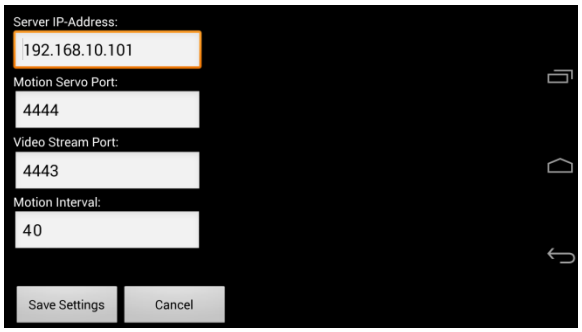
- IP-Adresse der Steuereinheit des Teleroboters (Server)
- Port der Head-Tracking Daten
- Port der Live-Videoübertragung

- Begrenzung der Anzahl der zu übertragenden Head-Tracking Informationen pro Sekunde

Die Anwendung kann in allen Views durch den Home-Button des Android-Gerätes beendet werden. Wird die Anwendung minimiert, werden das Abgreifen der Sensordaten des Smartphones und die Übertragung der Head-Tracking Daten unterbrochen.

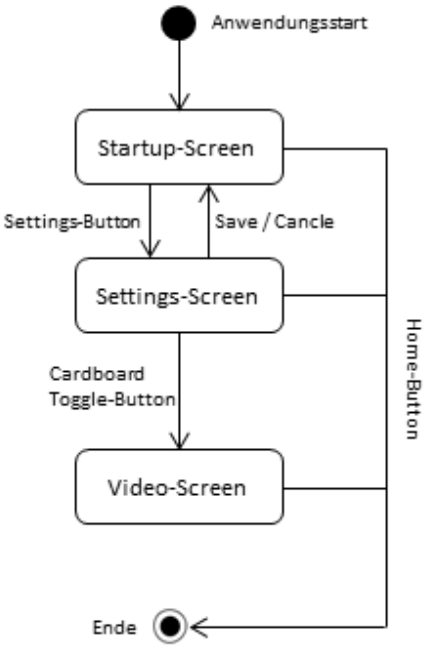


(a)



(b)

(c)



(d)

Abbildung 35: Views und Lebenszyklus der mobilen Anwendung

4.4.2 Steuereinheit des Teleroboters

Bei der Anwendung der Steuereinheit handelt es sich um eine Konsolenanwendung, die eine optionale grafische Benutzerschnittstelle zur Visualisierung von Statusinformationen zur Verfügung stellt. Die grafische Oberfläche basiert auf Swing. Dies ist eine von der Firma Sun Microsystems entwickelte Programmierschnittstelle und Grafikbibliothek zum Entwickeln von grafischen Benutzerschnittstellen.

Die grafische Benutzerschnittstelle der Steuereinheit ist optional, da die Software auf einem Embedded-System bzw. einem Einplatinencomputer, wie dem Raspberry Pi, laufen könnte, das keine Anzeige einer grafischen Schnittstelle ermöglicht.

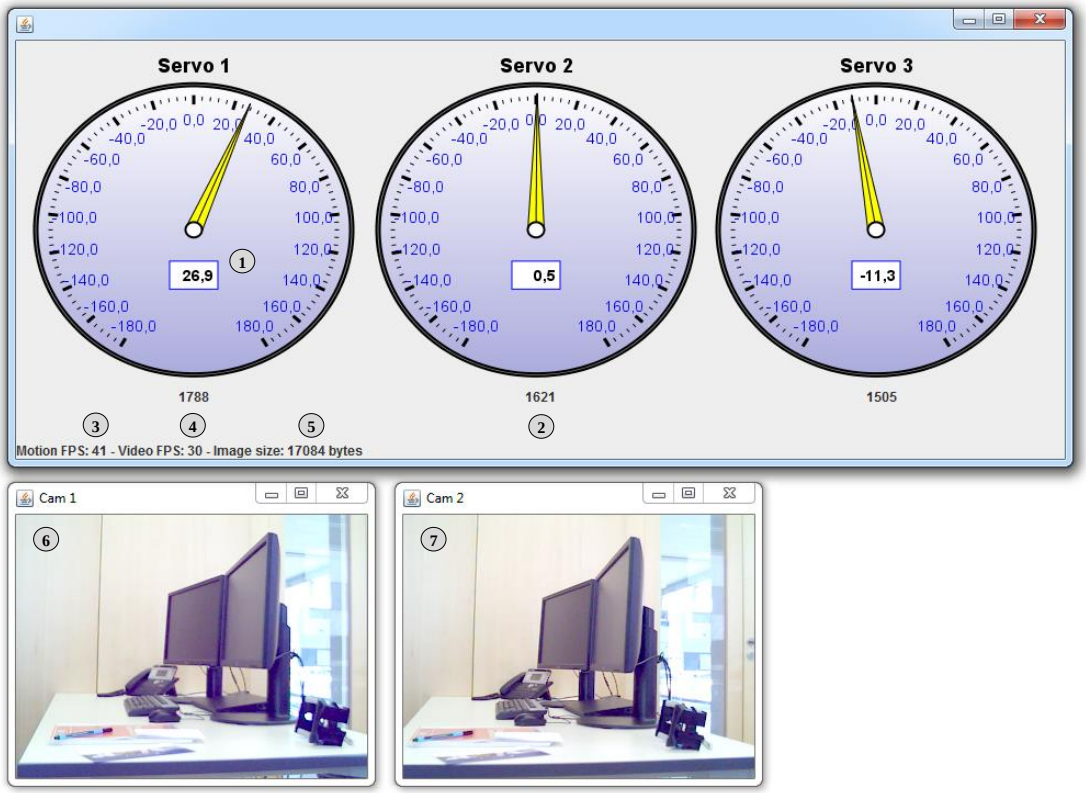


Abbildung 36: Grafische Benutzeroberfläche der Steuereinheit

Wird die Anwendung im sogenannten „GUI-Modus“ gestartet, können folgende grafische Informationen eingesehen werden:

- Stellung der drei Servomotoren des Teleroboters
- Performanceindikatoren der Übertragung von Video- und Head Tracking Daten
- Live-Videobild der zwei Augen des Teleroboters

Diese Informationen ermöglichen das einfache „Debuggen“ der internen Zustände des Teleroboters. Im Gegensatz zur reinen Konsolenausgabe ermöglicht die grafische Benutzerschnittstelle einen besseren Einblick in das Laufzeitverhalten der Anwendung. Abbildung 36 (S. 76) zeigt die Elemente der grafischen Benutzeroberfläche.

Das Hauptfenster stellt die aktuelle Position in Eulerwinkeln (Yaw, Pitch und Roll) der Servomotoren dar (1). Zu jedem Winkel wird die korrespondierende Position als Pulsdauer angezeigt (2), die zur Servosteuerung benötigt wird.

Die grafischen Diagramme zur Visualisierung der Positionen basieren auf der JFreeChart Bibliothek[10].

Der Status-Text in der Fußzeile des Hauptfensters gibt Informationen zur Performance der zwei Datenströme (4.3). Hierbei werden die Anzahl der empfangenen Positionsänderungen (Head-Tracking Nachrichten) pro Sekunde (3) und die Bildrate (4) der Videoübertragung (FPS) angezeigt. Die Datengröße des zu übertragenden Einzelbildes (5) verändert sich in Abhängigkeit der JPEG-Kompression. Diese Informationen ermöglichen eine Analyse des Datendurchsatzes der Datenströme.

Zusätzlich zum Hauptfenster werden zwei Fenster dargestellt, die das Kamerabild je Webcam beinhalten (6 – linkes Auge, 7 – rechtes Auge).

[10] http://www.jfree.org/jfreechart/samples.html

5. Validierung

Dieses Kapitel erläutert die Ergebnisse hinsichtlich der Ziele dieser Arbeit. Hierbei wird beschrieben, inwiefern die in 1.2 beschriebene Problemstellung gelöst werden konnte.

Die Performance der Datenübertragung und die Reaktionszeit der Visualisierung sowie die Steuerung der Kamera durch die Servomotoren sind die wichtigsten Elemente dieses Projektes. Wenn das Bild bei der Bewegung der Kamera ruckelt oder es den Anwender auch nur leicht zeitversetzt erreicht, kann dies das Realitätsgefühl minimieren. Der Anwender würde nicht das Gefühl bekommen, vor Ort zu sein und es könnte die sogenannte *Motion Sickness* (Kinetose) auftreten, bei der der Anwender unter Übelkeit leidet [VR01]. Das Gleiche gilt für die reaktionsschnelle Steuerung der Servomotoren.

5.1 Visualisierung

Die Latenzzeit der Visualisierung des Live-Videos, das die entfernte Umgebung darstellt, wird als akzeptabel wahrgenommen. Das Einzelbild je Auge hat allerdings eine sehr geringe Auflösung von nur *320 x 240* Pixeln und wird nach der Skalierung auf dem Smartphone-Display als „verpixelt“ wahrgenommen. Das Smartphone (Nexus 5), das zum Testen des Prototyps verwendet wurde, verfügt über eine ausreichend große Displayauflösung von *1920 x 1080* Pixeln. Allerdings ist die Minoru3D Webcam auf eine geringere Auflösung limitiert. Um das Realitätsgefühl zusätzlich deutlich zu steigern, sollte die Kamera die Full-HD Auflösung des Smartphones liefern, was einen Austausch der Kamera bedeuten würde. Da die Implementierung nicht von der verwendeten Kamera abhängig ist, kann diese bei einer möglichen Weiterentwicklung problemlos ausgetauscht werden.

Eine Steigerung der Bildauflösung würde allerdings aufgrund einer höheren zu übertragenden Datenmenge wiederum zu einer geringeren Latenzzeit bzw. Bildwiederholungsrate führen. Hier sollte ein angemessenes Mittel zwischen Latenzzeit und Bildqualität gefunden werden.

Für den Prototypen, der in dieser Arbeit entstanden ist, reicht die Minoru3D Webcam vollkommen aus, da die entfernte Umgebung durch die gute Latenzzeit der Videoübertragung in Echtzeit wahrgenommen wird und ein Immersions-Gefühl entsteht.

Auch die tonnenförmige Verzerrung zur Korrektur der Linsenkrümmung erhöht das Realitätsgefühl, da geradlinige Strukturen am Bildrand korrekt erscheinen. Allerdings wird das Bild auch ohne die Verzerrung als akzeptabel wahrgenommen, der Unterschied ist auf den ersten Blick nicht erkennbar. Dies scheint abhängig von den Linsen des verwendeten HMD zu sein. Der Verzerrungsalgorithmus reduziert die Bildrate um 10 Einzelbilder pro Sekunde.

5.2 Datenübertragung

Die Datenübertragung des Live-Videos stellt sich im Zuge des Projektes als Flaschenhals heraus, während die Übertragung der Head-Tracking Daten nicht ins Gewicht fällt. Um eine hohe Bildrate zu gewährleisten, ist ein starkes WLAN-Signal erforderlich.

Die Übertragung der Einzelbilder im JPEG-Format funktioniert erstaunlich gut und ermöglicht, bei einer stabilen WLAN-Verbindung, eine Bildrate von 40 Einzelbildern pro Sekunde (FPS). Das menschliche Auge nimmt eine Abfolge von Bildern ab einer Wiederholungsrate von ca. 14 bis 16 Einzelbildern pro Sekunde wahr, wobei ab einer Bildrate von 30 Bildern pro Sekunde ein Video als flüssig wahrgenommen wird [Vötter 2012].

Die Bildrate ist abhängig von der Kompression des JPEG-Bildes. Abbildung 37 zeigt das Verhältnis der Bildrate und dem Kompressionsfaktor bei der Komprimierung der Einzelbilder.

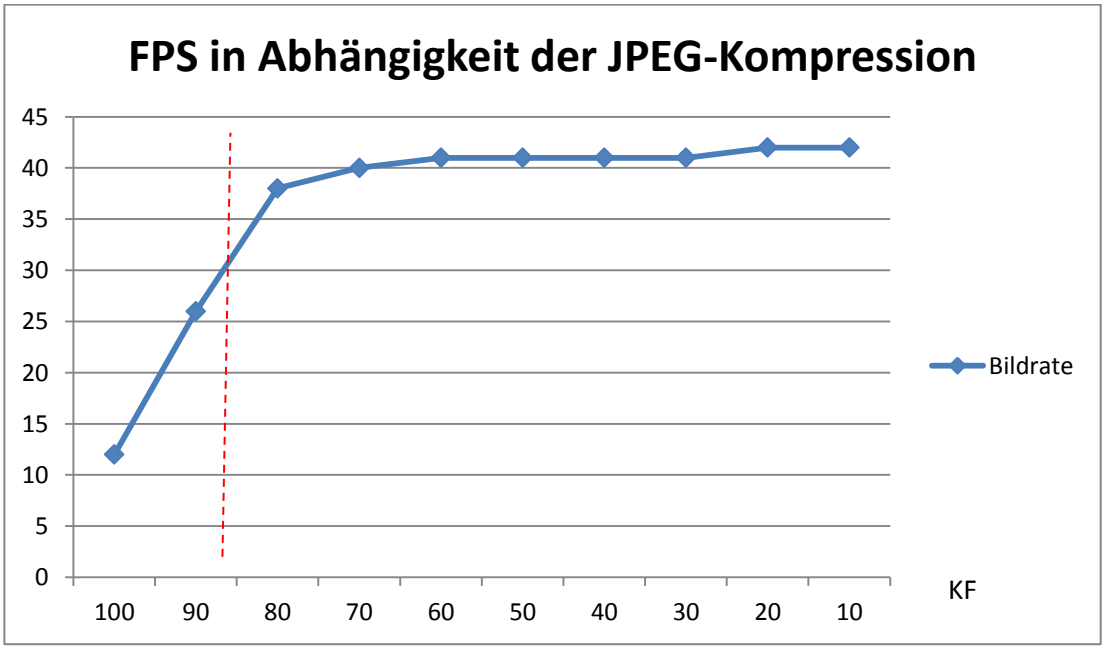


Abbildung 37: Bildrate in Abhängigkeit des Kompressionsfaktors der JPEG-Komprimierung

Dabei wurde ein Versuchsaufbau realisiert, bei dem ein Live-Video, bestehend aus Einzelbildern im JPEG-Format, unter realen Bedingungen und ohne eine Begrenzung der Bildrate, übertragen wurde. Da die verwendete Kamera eine Limitierung der Bildrate aufweist, wurde ein festes Standbild übertragen. Der Kompressionsfaktor wurde von 10% - 100% variiert und dabei die Bildrate auf dem Smartphone gemessen.

Der in Abbildung 37 (S.79) dargestellte Kompressionsfaktor wird durch einen Wert repräsentiert, der die Bildqualität in Prozent angibt. Ein Faktor von 100% beschreibt ein unkomprimiertes Bild. Geht der Faktor in Richtung 0%, wird das Bild mit der höchsten Kompressionsstufe und dem größten Qualitätsverlust komprimiert.

Ein Kompressionsfaktor von 50%, der zu einem komprimierten Bild mit einer noch akzeptablen Qualität führt, ermöglicht eine Bildrate von 41 Bildern pro Sekunde. Ein Einzelbild (*640 x 480* Pixeln) hat dabei eine Größe von *14.774* Bytes.

Bei einer höheren Bildqualität (Kompressionsfaktor von ca. 86%) müssen die Bildinformationen bei der Netzwerkübertragung auf zwei Nachrichten verteilt werden (gestrichelte Linie in der Abbildung). Dies verlangsamt die Übertragung des gesamten Bildes und reduziert die Bildrate.

Der Versuch zeigt, dass eine Übertragung ohne eine Begrenzung der Kamera eine viel höhere Bildrate erreichen könnte. Die Bildqualität verschlechtert sich drastisch ab einem Kompressionsfaktor von 40%. Ein höherer Faktor zeigt keinen essentiell wahrnehmbaren Qualitätsgewinn, so dass in dieser Arbeit die Wahl auf einen Kompressionsfaktor von 50% fiel.

5.3 Servosteuerung

Eine flüssige Bewegung der Servomotoren und eine schnelle Reaktion auf die Kopfbewegung des Anwenders hat einen signifikanteren Anteil an der Immersion.

Die preisgünstigen Hi_{TEC} HS-645MS Servomotoren verfügen über eine ausreichende Reaktionszeit und setzen die Bewegung der Kamera problemlos um. Das hohe Drehmoment der Motoren würde selbst ein schwereres Gewicht des Kameraaufbaus flüssig bewegen können.

In einem lokalen Netzwerk, mit einem stabilen WLAN-Signal, ermöglicht die Übertragung der Head-Tracking Daten, die auf die Servomotoren umgesetzt werden, einen Datendurchsatz von 40 Nachrichten pro Sekunde. Eine Nachricht enthält die drei

Winkel der HMD-Ausrichtung. Die Frequenz der Lageinformationen ist mehr als ausreichend.

Die Ansteuerung der Servomotoren ist sehr einfach über die serielle Schnittstelle zu realisieren.

5.4 Fazit

Das in dieser Arbeit entstandene System hat die wesentlichen Ziele erfüllt. Dabei ist ein funktionsfähiger Prototyp entstanden.

Der Anwender ist in der Lage, den Teleroboter mit der Kopfbewegung zu steuern, die durch das HMD aufgezeichnet wird. Die Steuerung erfolgt in Echtzeit und wird durch die geringe Latenz der Übertragung der Head-Tracking Daten ermöglicht. Die Umsetzung der Bewegung auf die Servomotoren erfolgt ohne Verzögerung.

Die Bildübertragung wird durch die JPEG-Kompression und den Einsatz des UDP-Protokolls als sehr flüssig und verzögerungsfrei wahrgenommen. Die geringe Bildauflösung, die durch die verwendete Minoru3D Kamera bedingt ist, lässt das Bild allerdings „pixelig“ erscheinen.

Die eingesetzte Hardware erfüllt die Anforderungen des Prototyps. Lediglich die Kamera sollte optimiert werden, indem sie in einer möglichen Weiterentwicklung durch ein besseres Produkt ausgetauscht werden könnte.

Durch die drahtlose Kommunikation per Teleroboter, kann dieser an jedem Ort, der über eine schnelle Netzwerk- bzw. Internetverbindung verfügt, eingesetzt werden.

Die Qualität der Sensoren eines herkömmlichen Smartphones reicht aus, um ein HMD zu realisieren und die Kopflage des Anwenders exakt zu bestimmen (getestet mit Nexus 5 und Samsung S3/4).

Das Google Cardboard dient als handliches Brillengestell und ist in Kombination mit einem Smartphone eine preisgünstige Alternative zu anderen VR-Brillen. Da das Cardboard aus Karton besteht, verschleißt es jedoch sehr schnell bei einer ständigen Benutzung.

6. Ausblick

Die Telepräsenz ist ein vielseitiges Gebiet, das auf ein großes Interesse bei Unternehmen und der Forschung stößt. Das zeigen zahlreiche innovative Produkte und Forschungsprojekte auf diesem Gebiet (2.15).

Nachdem die Ergebnisse dieser Arbeit eine erfolgreiche Realisierung eines Prototyps aufgezeigt haben, werden in diesem Abschnitt mögliche Erweiterungen und Optimierungen des entwickelten Systems aufgezeigt. Auch über diese Masterarbeit hinaus, sollte die Implementierung weiterentwickelt werden können. Aus diesem Grund wurde bei der Implementierung großen Wert auf die Erweiterbarkeit des Systems gelegt.

Das entwickelte System beschränkt sich auf die Übertragung der visuellen Wahrnehmung. Um die Immersion zu verbessern, könnte zusätzlich die auditive Wahrnehmung der entfernten Umgebung den Anwender erreichen. Ein weiterer Datenstrom, bestehend aus Audioinformationen, könnte dies umsetzen. Dies kommt bereits bei Videokonferenzsystemen zum Einsatz. Das Softwaredesign des entwickelten Systems ermöglicht die Erweiterung um einen weiteren Datenstrom, ohne eine zu große Änderung am System durchführen zu müssen.

Eine Optimierung der Videoübertragung wäre die dynamische Anpassung der Bildkompression in Abhängigkeit der Verbindungsqualität, um die bestmögliche Bildqualität bei unterschiedlichen Durchsatzbedingungen zu erreichen.

Die Servomotoren, welche die Kamera bewegen, und die Übertragung der Head-Tracking Daten weisen eine geringe Latenz auf. Um diese Reaktionszeit weiter zu reduzieren, könnte ein Ansatz verwendet werden, bei dem ein etwas größeres Sichtfeld, als das Head-mounted Display darstellen kann, aufgenommen wird. Dem Anwender wird jedoch ein kleinerer Ausschnitt präsentiert. Bewegt der Anwender seinen Kopf, wird das Sichtfeld im größeren Bild verschoben, bevor die Servomotoren reagieren können. Dieser Ansatz wurde von [NLieb] vorgestellt (2.15).

Für den produktiven Einsatz des Systems sollte der Sicherheitsaspekt berücksichtigt werden. Eine Ende-zu-Ende-Verschlüsselung wäre je Anwendungsfall sinnvoll, um die Vertraulichkeit und die Integrität der Datenübertragung zu gewährleisten. Ein Sicherheitskonzept könnte die Manipulation des Systems verhindern, würde jedoch die Performance der Übertragung verschlechtern. Bei dem dieser Arbeit zugrunde

liegenden Prototypen wurde eine Verschlüsselung der Datenübertragung nicht berücksichtigt.

Das System könnte auch mit anderen Technologien kombiniert werden. Hierbei gibt es viele Möglichkeiten, das Projekt zu erweitern. Ein Greifarm des Roboters könnte Aufgaben in der entfernten Umgebung durchführen. Haptische Schnittstellen wie Handschuhe könnten die Tastempfindungen des Greifarms zurück an den Anwender in Form von Druck oder Vibrationen senden. Eine weitere Möglichkeit wäre eine Montage auf einem fahrbaren Untersatz. Der Teleroboter wäre beweglich und nicht auf einen Standort beschränkt.

An der Hochschule Offenburg könnte das entwickelte System bei der Besichtigung des Campusgeländes oder der Labore eingesetzt werden, bei dem der Benutzer sein Smartphone als HMD verwendet.

Anhang

1. Beigefügte CD

Die beigefügte CD enthält die Software des entwickelten Prototyps:

- Quellcode der Anwendung (Steuereinheit und Smartphone-Anwendung)
- Sämtliche verwendete Java-Bibliotheken
- Die Anwendung als ausführbare .jar-Dateien

2. JPEG-Stream Performance Test

Folgende Tabelle zeigt die Performance der Videoübertragung, gemessen in Einzelbilder pro Sekunde (Framerate):

Bildqualität	**Einzelbildgröße in Bytes**	**Framerate**
10%	6.578	42
20%	9.093	42
30%	11.310	41
40%	13.067	41
50%	14.774	41
60%	16.599	41
70%	19.387	40
80%	24.039	38
90%	35.056	26
100%	111.039	12

3. Verbaute Hardware

Der Teleroboter besteht aus folgenden Hardwareteilen:

Produkt	Preis	Quelle
AL5A Robotic Arm Combo Kit	220 €	lynxmotion.com
3 x HS-645MG Servomotor	29 € je Motor	lynxmotion.com
SSC-32 Servo Controller	37 €	robotshop.com

4. Vergleich der VR-Brillen

Produkt	**Auflösung**	**Framerate**	**Sichtfeld**	**Gewicht (Gramm)**	**Preis**
Silicon Micro Display	1920x1080	240	45°	180	799 USD
Oculus Rift	1280x800	60	110°	369	300 USD
Carl Zeiss Cinemizer	1280x720, 1920x1080, HDMI 3D		30°	120	635 EUR
Sony HMZ-T2	1280x720	24	45°	330	799 EUR
Sony HMZ-T3W	1280x720	24	45°	320	1.299 EUR
Glyph	1280x720 je Auge	120	45°	450	500 USD
Epson Moverio – BT-100	960x540		23°	240	499 EUR
Durovis Dive	je nach verwendetem Smartphone			150	57 EUR
Valve Prototyp HMD	2160×1280	-	-	-	-
CastAR	720p je Auge	-	90	< 100	ab 200 USD
Infinite Eye	1280 x 800 Pixel je Auge	-	210	300	-
Sulon GVX	1920x1080	-	160	-	ca. 500 USD
Sony Morpheus	1920x1080	60	90°	-	-
Samsung Gear VR	je nach verwendetem Smartphone			555	199 USD
Avegant Glyph	1.280 x 720		45°	850	499 USD
Oculus Rift Crescent Bay	mehr als 1920x1080	60		-	-
Google Cardboard VR	je nach verwendetem Smartphone				20 USD
Vrvana Totem	1080p	75	90°	400	350 EUR

5. Nexus 5

Folgende Tabelle zeigt die Spezifikationen des Smartphones (Nexus 5), das während der Entwicklung zum Einsatz kam:

Hersteller	Google
Abmessungen (HxBxT)	137,84 x 69,17 x 8,59 mm
Gewicht	130 Gramm
Gerätefarben	Schwarz, Weiß, Rot
Marktstart	Nov 13
Google Nexus 5 16 GB	349 Euro
Google Nexus 5 32 GB	399 Euro
Betriebssystem	Android OS 4+
Akku - Typ	Li-Polymer
Akkuleistung	2300 mAh
Standby	300 Stunden
Displaytyp	IPS-LCD
Displaydiagonale	4,95 Zoll (12,57 cm)
Displayauflösung	1920 x 1080 Pixel
Pixeldichte	445 ppi (Pixel/Zoll)
Chipsatz	Snapdragon 800
Anzahl Kerne	4 x 2300 Megahertz
Grafik-Chip	Adreno 330
Arbeitsspeicher / RAM	2048 MB
Speicher (intern)	32000 MB
Sensoren	• Annäherungssensor • Digitaler Kompass • Gyroskop • Lagesensor • Lichtsensor

Literaturverzeichnis

[Jehle 2014] Martin Jehle, Frédéric Starnecker; “Moved Reality”; HTWG-Konstanz 2014.

[CT2014] c’t 2014, Heft 18; „Fremdwahrnehmung“.

[Koller u. a. 1997] Dieter Koller u. a.; „Real-time Vision-Based Camera Tracking for Augmented Reality Applications“; Symposium on Virtual Reality Software and Technology 1997.

[Forbes] Forbes 08/2014; “Could Virtual Reality Be The Next Big Thing In Education?”.

[ComScore 2014] „http://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenutzer-in-deutschland-seit-2010/“; aufgerufen am: 01.03.2015.

[Lawitzki 2012] Paul Lawitzki; „Application of Dynamic Binaural Signals in Acoustic Games“; Hochschule der Medien Stuttgart 2012.

[Rößler 2009] Patrick Rößler; “Telepräsente Bewegung und haptische Interaktion in ausgedehnten entfernten Umgebungen”; Dissertation Universität Karlsruhe 2009.

[Schünkle 2000] Michael Schünkle; „Topografie und Funktion des Bewegungssystems“, Georg Thieme Verlag 2000; ISBN: 978-3131185716

[MS 2014] „http://support.microsoft.com/de-de/kb/822061“; aufgerufen am: 02.04.2015

[SSC32] SSC32 Manual Ver. 2.0; Lynxmotion

[Dodgson 2004] Neil A. Dodgson; “Variation and extrema of human interpupillary distance”; University of Cambridge 2004

[Berestesky u. a. 2004]	Berestesky, Chopra, Spong, Mark; „Discrete Time Passivity in Bilateral Teleoperation over the Internet“; IEEE Conference on Robotics and Automation 2004.
[TUB2012]	H.264 Award; Pressestelle TU Berlin; Januar 2012
[Grabowski 2014]	H. Grabowski; „Mobile Computing“; HS Offenburg 2014
[NLieb]	N. Lieb; “Fly Like a Bird”; Kickstarter Projekt
[Azuma 1997]	Ronald T. Azuma ; “A Survey of Augmented Reality”; August 1997
[Slater 2003]	Mel Slater; “A Note on Presence Terminology”; University College London 2003
[Schwan u. Buder 2006]	Stephan Schwan, Jürgen Buder; “Virtuelle Realität und E-Learning”; 24.3.2006
[Gartner 2015]	„http://de.statista.com/statistik/daten/studie/12885/umfrage/marktanteil-bei-smartphones-nach-betriebssystem-weltweit-seit-2009/“, aufgerufen am: 22.04.2015
[Schomburg 2013]	Constantin Schomburg; „Evaluierung einer Sensor-Simulationsumgebung zum Testen von Android-Anwendungen“; Universität Hannover 2013
[Wiese 2014]	Hendrik Wiese; „Entwicklung einer Basisplattform für Telepräsenzsysteme“; 2014
[Wu 2013]	Wenzhuo Wu; „Taxel-Addressable Matrix of Vertical-Nanowire Piezotronic Transistors for Active and Adaptive Tactile Imaging“; 2013
[Muszynski 2012]	Sebastian Muszynski; “Teleoperation eines Service-Roboters mit Android-Handhelds”; 2012

[Vötter 2012] Raffael Vötter, PC Games 2012; “http://www.pcgameshardware.de/Spiele-Thema-239104/Specials/Wann-laufen-Spiele-fluessig-1034704”, aufgerufen am: 10.04.2015

[iRobot] “http://www.irobot.com/For-Business/Platform-Opportunities.aspx”, aufgerufen am: 17.04.2015

[RP7i] “http://www.intouchhealth.com/products-and-services/products/rp-7i-robot/”, aufgerufen am: 17.04.2015

[HTC] “ http://www.htcinside.de/nexus-one/datenblatt/”, aufgerufen am: 05.04.2015

[Google01] Google Developer Reference; „http://developer.android.com/reference/android/hardware/SensorEvent.html“; aufgerufen am: 24.04.2015

[Google02] “http://web.archive.org/web/20090228170042/http:/source.android.com/posts/opensource”; aufgerufen am: 24.04.2015

[OpenCV] “http://www.opencv.org”; aufgerufen am: 25.03.2015

[JWPlayer] „The Pain of Live Streaming on Android“; „http://www.jwplayer.com/blog/the-pain-of-live-streaming-on-android “; aufgerufen am: 08.02.2015

[Bosch] „http://www.bosch-sensortec.com /en/homepage/products_3/3_axis_sensors/acceleration_sensors/bma150_4/bma150“; aufgerufen am: 24.04.2015

[Jpeg] „http://www.jpeg.org“; aufgerufen am: 22.04.2015

[VR01]	„http://www.onlyvr.de/virtuelle-realitaet/gefahren“; aufgerufen am: 15.03.2015
[SM01]	„http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/Google-Needs-a-Strategy-for-Video-on-Android-Devices-86846.aspx“; aufgerufen am: 24.04.2015
[AML01]	„http://lists.apple.com/archives/quartzcomposer-dev/2011/May/msg00053.html“; aufgerufen am: 05.01.2015
[Sherman u. Craig 2000]	Sherman, Craig; „Understanding Virtual Reality: Interface, Application, and Design“; 2000; ISBN: 978-1558603530
[Blanken 2011]	Malte B. Blanken; “Mobile Augmented Reality: Neue Möglichkeiten im Bereich der visuellen Kommunikation”; 2011